Руководство по настройке серверов и настольных систем для любого дистрибутива

5-е издание
исправлено и дополнено

ЗАПУСКАЕМ LINUX

Символ®

O'REILLY®

Маттиас Калле Далхаймер и Мэтт Уэлш

RUNNING LINUX

Fifth Edition

Matthias Kalle Dalheimer
and Matt Welsh

O'REILLY®

ЗАПУСКАЕМ LINUX

Пятое издание

Маттиас Калле Далхаймер
и Мэтт Уэлш

Санкт-Петербург — Москва
2008

Маттиас Калле Далхаймер и Мэтт Уэлш

Запускаем Linux, 5-е издание

Перевод А. Киселева

Главный редактор	*А. Галунов*
Зав. редакцией	*Н. Макарова*
Научный редактор	*О. Цилюрик*
Редактор	*Е. Бочкарева*
Корректор	*С. Минин*
Верстка	*Д. Орлова*

Далхаймер К., Уэлш М.
Запускаем Linux, 5-е издание. – Пер. с англ. – СПб.: Символ-Плюс, 2008. – 992 с., ил.
ISBN-10: 5-93286-100-2
ISBN-13: 978-5-93286-100-4

Классический труд, вышедший уже 5-м изданием, значительно расширен с целью отразить зрелость операционной системы и изобилие существующего ПО. Такие горячие темы, как воспроизведение звука и видео, ПО для рабочих групп и фильтрация спама, соседствуют рядом с основами настройки и администрирования, которые всегда делали книгу популярной. Издание охватывает не только основные способы обмена информацией (электронная почта, навигация в Сети и обмен мгновенными сообщениями), но и описывает тонкости настройки сети, включая соединение по коммутируемым линиям, ADSL и кабельные модемы. Перечень новых тем включает шифрование электронной почты и шифрующие файловые системы, передовые методы работы с командными оболочками и приложения удаленной регистрации. Классические темы, касающиеся загрузки, управления пакетами, пересборки ядра и настройки X Window, также обновлены. Предвидя возможные трудности, авторы предлагают надежные решения и дают четкие и ясные инструкции, которые обеспечат вам успешную работу в Linux. Материал излагается просто и ясно, но представлен достаточно полно, чтобы служить руководством для начинающих и в то же время обеспечить новой информацией опытных пользователей, стремящихся побольше узнать о Linux.

ISBN–10: 5–93286–100–2
ISBN–13: 978–5–93286–100–4
ISBN 0–596–00760–4 (англ)

Издательство «Символ-Плюс». 199034, Санкт-Петербург, 16 линия, 7,
тел. (812) 324-5353, www.symbol.ru. Лицензия ЛП N 000054 от 25.12.98.
Налоговая льгота – общероссийский классификатор продукции
ОК 005-93, том 2; 953000 – книги и брошюры.
Подписано в печать 21.12.2008. Формат 70x100$^{1}/_{16}$. Печать офсетная.
Объем 62 печ. л. Тираж 2000 экз. Заказ N
Отпечатано с готовых диапозитивов в ГУП «Типография «Наука»
199034, Санкт-Петербург, 9 линия, 12.

Оглавление

Предисловие

Недостаточно овладеть техникой. Нужно превзойти технические приемы, и тогда искусство станет естественным, проистекая из подсознания.

Дайсецу Судзуки (Daisetsu Suzuki) (1870–1966)

Эта книга рассказывает об операционной системе Linux, бесплатной и распространяемой с открытыми исходными текстами (open source), благодаря которой происходят изменения в мире компьютеров. В этой книге показывается, что, изучив эту мощную бесплатную операционную систему, можно начать работать на компьютере совершенно по-иному. Разрабатываемая свободно образованной группой, состоящей из тысяч добровольцев, связанных через Интернет, Linux противоречит традиционному компьютерному «мейнстриму». Linux зародилась как настоящее подпольное движение, можно сказать, хакеров-партизан и в значительной мере вернула атмосферу радости, открытий и самостоятельности в культуру вычислительной техники, в которой в настоящее время доминируют корпорации. Мы приглашаем вас погрузиться в этот мир, получить удовольствие и присоединиться к той массе людей, которые понимают, что значит управлять синхросигналом дисплея и устанавливать корневой раздел в образе ядра с помощью *rdev*.

Цитата из Дзэн, приведенная в начале, подытоживает философию, которой мы придерживаемся в этой книге, рассчитанной на пытливого и творческого читателя, готового с головой окунуться в мир Linux и желающего разобраться в самых основах этой системы. Linux – это восстание против коммерческих операционных систем, и многие ее пользователи принадлежат к тому типу людей, которые любят находиться на переднем крае технологических новшеств. Разумеется, любой читатель может без особых хлопот установить и запустить систему Linux, но цель этой книги – помочь глубже вникнуть в систему и позволить полностью овладеть характерным для Linux стилем мышления. Вместо того чтобы вдаваться в подробности запутанных моментов, мы постараемся объяснить принципы работы системы, чтобы читатель мог самостоятельно решать возникающие проблемы. Предоставляя в общее распоряжение опыт, накопленный несколькими специалистами по Linux, мы надеемся вселить в вас достаточную уверенность, чтобы вы могли назвать себя настоящим гуру Linux. (Вот ваш первый коан: какой смысл заниматься хакерством в одиночку?)

У вас в руках пятое издание «Запускаем Linux» – классической, по общему мнению, книги по установке, сопровождению и изучению системы Linux. Первое издание (на англ. языке) вышло еще в 1996 году и ведет свое происхождение от свободно распространявшейся книги «Linux Installation and Getting Started»

Мэтта Уэлша (Matt Welsh), которую все еще можно найти в Интернете. С тех пор книга подверглась ряду улучшений и изменений с целью привести ее в соответствие с последними разработками в мире Linux.

Калле Далхаймер (Kalle Dalheimer), разработчик и консультант, обладающий огромным опытом в области разработки системного и прикладного программного обеспечения для Linux, был одним из лидеров при работе над последними тремя изданиями. Длительное время существенную помощь оказывали: Лар Кауфман (Lar Kaufman) (подготовка материалов к печати), Том Аделстейн (Tom Adelstein) (он взялся обновить вступительную главу и добавил материалы о VMWare, rdesktop, VNC и FreeNX), Аарон Вебер (Aaron Weber) (GNOME, Evolution, Red Carpet и ZENWorks), Сэм Хайзер (Sam Hiser) (OpenOffice), Джей Тс (Jay Ts) (SAMBA), Джон Терпстра (John H. Terpstra) (обновление пакета SAMBA и NFS), Джефф Трантер (Jeff Tranter) (мультимедиа, источники информации о Linux), Кайл Ранкин (Kyle Rankin) (игры), Брекин Логгинс (Breckin Loggins) (GnuCash), Род Смит (Rod Smith) (сведения о системе печати, включая CUPS), Кайл Дент (Kyle Dent) (Postfix), Терри Доусон (Terry Dawson) (сведения о безопасности), Брайан Винсент (Brian Vincent) (Wine и CodeWeaver), Крис Лоуренс (Chris Lawrence) (система пакетов Debian), Ватафу Валерик (Vatafu Valerica) (глава о LAMP), Марк Муц (Marc Mutz) (сведения о шифровании с открытым ключом и о шифрующих файловых системах), Стефен Хансен (Stefen Hansen) (информация о GIMP, OpenGL, Postfix и ProFTPd), Тилл Адам (Till Adam) (материал о существующих решениях для рабочих групп), Джеспер Педерсен (Jesper Pedersen) (информация о kimdaba и Procmail, ряд дополнений в разделе, рассказывающем о языке программирования Python), Михаэль Бойер де ла Жироде (Michael Boyer de la Giroday) (PHP), Айван Ристик (Ivan Ristic) (обновления в разделах, рассказывающих о веб-сервере Apache и LAMP) и Джеффри Дуниц (Jeffrey Dunitz) (дополнения к главе, рассказывающей о резервном копировании).

По мере своего развития операционная система Linux становится все более привлекательной для использования в новых областях науки и техники, что являет прекрасный повод для создания книги, такой как эта, которая способствовала бы продвижению Linux, с каждым годом обретающей все новые возможности. Это издание значительно превосходит по объему все предыдущие издания и гораздо глубже охватывает темы, как, например, инструментальные средства рабочего стола, которые ранее упоминались лишь ради приличия. Ни одна книга не в состоянии вместить всю информацию, которую необходимо знать о Linux, поэтому всякий раз мы задавались вопросом, какая информация будет представлять наибольшую ценность для тех, кто изучает новую для себя систему и пытается получить основные сведения о ней, на основе которых они смогут двигаться дальше. Наш способ изложения материала прекрасно зарекомендовал себя в прошлых изданиях, и мы полагаем, что данная книга еще долгое время будет служить вам верой и правдой.

В предисловии к первому изданию говорилось о том, что у Linux есть все возможности полностью изменить мир операционных систем для PC. Теперь мы можем с полной уверенностью сказать, что это предсказание оправдалось! Linux с поразительным напором вторглась в центральное русло развития вычислительной техники: о ней рассказывают все крупные средства массовой информации, она дала толчок так называемой революции open source (разработке программного обеспечения с открытым исходным кодом) и многими рассматривается как наиболее

сильный конкурент, способный противостоять засилью Microsoft на рынке операционных систем. Большинство оценок сходится на том, что количество пользователей Linux в мире превысило 300 миллионов. В своем развитии Linux достигла той степени зрелости, когда большинству пользователей стало не обязательно разбираться в сложностях драйверов устройств, файлах настроек XFree86 и начальных загрузчиках, чтобы начать использовать эту операционную систему. Фактически установка современных дистрибутивов Linux стала таким же простым делом, что и установка коммерческих систем, подобных Microsoft Windows. Тем не менее мы считаем целесообразным дать читателю некоторое представление о внутреннем устройстве системы, чтобы он мог понять механизм ее работы, даже несмотря на то, что для обычной работы с Linux в этом нет строгой необходимости.

Структура этой книги

Каждая глава этой книги содержит изрядную порцию информации. Полное подробное изложение материала могло бы занять несколько книг, но мы быстро пройдем темы, которые необходимо знать.

Первая часть книги, «Удобство и производительность Linux», продемонстрирует возможности операционной системы с точки зрения рядового пользователя. Здесь будет рассказываться о том, как работать с электронной почтой, о веб-серфинге, об играх, как просматривать видеофильмы, и тому подобное.

Глава 1 «Введение в Linux»

Рассказывается сразу о многом. Здесь объясняется история появления Linux и что она может предложить, чтобы привлечь к себе внимание новых пользователей.

Глава 2 «Подготовка к установке и установка»

Описываются предварительные действия, которые необходимо выполнить перед установкой, например создание разделов на жестком диске, и последующие шаги по установке и начальной настройке Linux.

Глава 3 «Окружение рабочего стола»

Поможет настроить рабочий стол и ряд основных программных продуктов, таких как утилита для работы с электронной почтой Evolution, календарь и органайзер, наиболее комфортабельным образом.

Глава 4 «Основы командной строки UNIX»

Представляет собой введение в UNIX для системного администратора и предназначена тем, кто в этом введении нуждается. Данная глава содержит информацию, необходимую для решения основных задач, которые будут возникать на протяжении всей книги. В этой главе рассказывается об основных командах и некоторых понятиях, которые необходимо знать, а также даются полезные советы администраторам.

Глава 5 «Веб-броузеры и обмен мгновенными сообщениями»

Демонстрируются наиболее интересные приемы, используемые в таких популярных видах деятельности, как путешествие по Интернету и обмен мгновенными сообщениями.

Глава 6 «Клиенты электронной почты»

Рассказывается о других программах, предназначенных для работы с электронной почтой, для тех, кто не желает ограничивать себя использованием Evolution, а также демонстрируются возможные приемы повышения уровня безопасности при работе с электронной почтой.

Глава 7 «Игры»

Описывается внушительное количество игр, как автономных, так и сетевых, поддерживаемых операционной системой Linux.

Глава 8 «Офисные пакеты и приложения личного пользования»

Рассказывается о том, что работа в офисе под управлением Linux может быть такой же простой, как и при использовании коммерческих аналогов. В основном обсуждение коснется офисных пакетов OpenOffice и KOffice и приложения для решения финансовых задач GnuCash. Кроме того, здесь есть вводная информация об инструментальных средствах для рабочих групп.

Глава 9 «Мультимедиа»

Обсуждаются вопросы прослушивания аудио- и просмотра видеофайлов. Рассмотрены основные понятия, которые будут полезны при настройке системы, где отсутствуют инструментальные средства автоматической настройки, а также рассказывается о некоторых наиболее известных приложениях. Здесь же приводятся основы работы с графическим редактором GIMP.

Вторая часть книги, «Системное администрирование», описывает порядок настройки операционной системы Linux и программного окружения для выполнения таких задач, как печать и использование файлов совместно с другими системами в сети. Здесь также описываются некоторые из способов обслуживания операционной системы.

Глава 10 «Основы системного администрирования»

Охватываются темы, связанные с администрированием, такие как файловые системы и разделы для свопинга, которые обычно создаются автоматически в процессе установки, но иногда их обслуживание требует вмешательства пользователя.

Глава 11 «Управление пользователями, группами и привилегиями»

Охватывает такие фундаментальные темы, как управление пользователями и правами доступа (привилегиями), которые служат строительными блоками системы безопасности Linux.

Глава 12 «Установка, обновление и сборка программ»

Описываются принципы обновления системы, что является немаловажным фактором, влияющим как на приобретение новых функциональных возможностей, так и на ликвидацию проблем, связанных с системой безопасности.

Глава 13 «Подключение к сети»

Представляет собой введение в настройку сетевого окружения, которое обычно выполняется в процессе установки, но которое следует знать на более глубоком уровне. В главе рассказывается, как настроить систему для работы в локальной сети и как подключить компьютер к Интернету с помощью про-

токола PPP (Point-to-Point Protocol – протокол «точка-точка»). Кроме того, здесь же обсуждаются вопросы настройки ISDN и ADSL.

Глава 14 «Система печати»

Демонстрируется, как научить операционную систему Linux распознавать принтеры и как управлять печатью документов.

Глава 15 «Совместное использование файлов»

Рассматривает вопросы совместного использования файлов в сети, фокусируясь в основном на использовании пакета программного обеспечения Samba, который позволяет использовать файлы и принтеры совместно с операционными системами Windows.

Глава 16 «X Window System»

Описывается порядок настройки X Window System, которая лежит в основе окружения рабочего стола, рассмотренного в главе 3. Здесь рассказывается, как преодолеть проблемы, с которыми можно столкнуться в процессе установки, и как добиться максимальной производительности от видеокарты.

Глава 17 «Запуск и остановка системы»

Описывает порядок запуска и остановки системы. Среди всего прочего в главе описывается порядок установки и обслуживания загрузчика GRUB, который предоставляет возможность выбора загрузки одной из нескольких операционных систем.

Глава 18 «Настройка и сборка ядра»

Описывает порядок обновления ядра и его модулей, что может потребоваться для расширения возможностей операционной системы или для установки драйверов имеющегося аппаратного обеспечения.

Третья часть книги, «Программирование», начинается с описания такой интересной темы, как программирование, которая еще больше повышает привлекательность Linux.

Глава 19 «Текстовые редакторы»

Представляет собой детальное описание замечательных текстовых редакторов *vi* и Emacs. Здесь же описываются некоторые дополнительные возможности по обработке и форматированию текстовых документов, что может служить достойной альтернативой использованию текстовых процессоров.

Глава 20 «Обработка текстовых документов»

В этой главе описываются инструментальные средства, позволяющие получать прекрасно оформленные документы из исходных текстов, созданных с использованием языков разметки, таких как XML, TeX, troff и Texinfo.

Глава 21 «Инструментальные средства программиста»

Представляет собой обширное введение в тему программирования для операционной системы Linux. Здесь дается краткое описание множества языков программирования, а также инструментальных средств, знание которых вам наверняка пригодится, даже если вы далеки от программирования.

Четвертая часть книги, «Сетевые службы», описывает ряд сетевых служб и некоторые более подробные сведения, связанные с работой в сетях.

Глава 22 «Запуск веб-сервера»

Описывает процесс установки и настройки веб-сервера Apache – одного из самых популярных веб-серверов в мире.

Глава 23 «Транспортировка и обработка сообщений электронной почты»

Описывает простой в использовании почтовый сервер Postfix и некоторые другие инструментальные средства, как, например, SpamAssassin.

Глава 24 «Запуск сервера FTP»

Описывает безопасный способ предоставить файлы, которые можно будет загрузить с других компьютеров.

Глава 25 «Запуск веб-приложений с использованием MySQL и PHP»

Описывает символы M и P широко известной аббревиатуры LAMP. Рассказывает о настройке и подготовке MySQL и PHP для совместной работы с сервером Apache.

Глава 26 «Система безопасности»

Описывает веб-сервер ProFTPd, который может с успехом использоваться для организации совместного доступа к документам между коллегами или для широкой общественности.

Глава 27 «Резервное копирование и восстановление»

Основные приемы по обеспечению сохранности критически важных данных.

Глава 28 «Работа в гетерогенных сетях и запуск программ Windows»

Описывает способы, с помощью которых можно извлечь максимум пользы от применения двух таких разных операционных систем.

Приложение A «Источники информации по Linux»

Рассказывается о наиболее интересных ресурсах Интернета, где можно найти документацию с описанием Linux.

Способы оформления, используемые в книге

Ниже приводится список обозначений, принятых в этой книге:

Курсив

Используется для обозначения имен файлов и каталогов, названий программ и команд, параметров командной строки, адресов электронной почты и имен сайтов, а также при определении новых терминов.

`Моноширинный шрифт`

Используется в примерах для отображения содержимого файлов, данных, возвращаемых командами, обозначения переменных окружения и ключевых слов в программном коде, а также команд Emacs.

`Моноширинный полужирный шрифт`

Используется в примерах для выделения команд или другого текста, которые пользователь должен ввести.

Моноширинный курсив

Используется для обозначения переменных параметров, ключевых слов или текста, которые пользователь должен заменить фактическим значением.

Этим значком отмечены важные примечания к тексту, расположенному рядом.

Этим значком отмечены предупреждения, относящиеся к тексту, расположенному рядом.

Использование примеров программного кода

Данная книга предназначена оказать вам помощь в решении ваших повседневных задач. Вы можете беспрепятственно использовать программный код, приводимый в книге, в своих программах или в документации. Вам не нужно обращаться в издательство за получением разрешения на его использование при условии, что вы не собираетесь воспроизводить существенные по объему участки программного кода. Например, если при написании своей собственной программы вы использовали некоторые небольшие участки, взятые из примеров, приводимых в книге, вам не нужно обращаться за разрешением. Однако, если вы собираетесь перепродавать или распространять компакт-диск с примерами из книг, выпущенных издательством O'Reilly, тогда вам необходимо получить соответствующие разрешения. Если при ответе на чьи-либо вопросы вы собираетесь процитировать некоторые выдержки из этой книги или продемонстрировать отрывок из примера, вам не нужно обращаться за разрешением. Однако, если вы предполагаете включить в свою документацию существенные объемы программного кода примеров из этой книги, вам следует обратиться за разрешением.

Мы признательны за указание авторства, но не требуем этого. Обычно указание источника включает название, автора, издателя и ISBN. Например: «*Running Linux*, Fifth Edition by Matthias Kalle Dalheimer and Matt Welsh. Copyright 2006 O'Reilly Media, Inc., 0-596-00760-4».

Если у вас возникнут сомнения по поводу законности использования программного кода примеров из книги, обратитесь за разъяснениями по адресу *permissions@oreilly.com*.

Как с нами связаться

Информация, приведенная в данной книге, была со всей тщательностью протестирована и проверена, но вы можете обнаружить некоторые технические неточности (или даже ошибки!). Пожалуйста, сообщайте нам об обнаруженных ошибках, а также присылайте ваши предложения относительно будущих изданий по адресу:

O'Reilly & Associates, Inc.
1005 Gravenstein Highway North
Sebastopol, CA 95472

800-998-9938 (в США или Канаде)
707-829-0515 (международный или местный)
707-829-0104 (факс)

Можно писать по электронной почте. Для включения в список рассылки по электронной почте или заказа каталога пишите по адресу:

info@oreilly.com

С техническими вопросами и комментариями по поводу книги можно обратиться по адресу:

bookquestions@oreilly.com

У этой книги есть веб-сайт, на котором представлены примеры, найденные ошибки и планы по выходу новых изданий. Эта страница находится по адресу:

http://www.oreilly.com/catalog/runlinux5

Дополнительные сведения об этой и других книгах можно найти на веб-сайте O'Reilly & Associates:

http://www.oreilly.com/

Safari® Enabled

Если на обложке технической книги есть пиктограмма «Safari® Enabled», это означает, что книга доступна в Сети через O'Reilly Network Safari Bookshelf.

Safari предлагает намного лучшее решение, чем электронные книги. Это виртуальная библиотека, позволяющая без труда находить тысячи лучших технических книг, вырезать и вставлять примеры кода, загружать главы и находить быстрые ответы, когда требуется наиболее верная и свежая информация. Она свободно доступна по адресу *http://safari.oreilly.com*.

Благодарности

Эта книга – результат работы многих людей и, по-видимому, всех их перечислить здесь невозможно. Прежде всего, мы хотим поблагодарить Энди Орама (Andy Oram), который проделал отличную работу по редактированию и щелканью кнутом, благодаря чему книга приобрела достойный вид. Помимо общего руководства Энди написал главу с руководством по UNIX, а также разделы, описывающие X, Perl и Gaim. Именно Энди первым предложил написать книгу для O'Reilly, и проявил долготерпение святого в ожидании наших исправлений и дополнений.

Объем книги постоянно увеличивался, а обсуждаемые в ней темы оказались настолько разнообразными, что один человек не смог охватить их все (как, впрочем, и небольшой коллектив соавторов). Поэтому к освещению существенной доли материала мы привлекли большое число экспертов в самых разных областях, имена которых мы уже упоминали в самом начале предисловия.

Мы хотим также поблагодарить следующих людей за работу над операционной системой Linux (без них не о чем было бы писать): Линуса Торвальдса (Linus Torvalds), Ричарда Столлмена (Richard Stallman), Дональда Беккера (Donald Beck-

er), Алана Кокса (Alan Cox), Реми Карда (Remy Card), Эрика Реймонда (Eric Raymond), Теда Тсо (Ted T'so), Х. Дж. Лю (H. J. Lu), Мигеля де Иказу (Miguel de Icaza), Росса Биро (Ross Biro), Дрю Экхардта (Drew Eckhardt), Эда Карпа (Ed Carp), Эрика Янгдейла (Eric Youngdale), Фреда ван Кемпена (Fred van Kempen), Стивена Твиди (Steven Tweedie), Патрика Волькердинга (Patrick Volkerding), Дирка Хондела (Dirk Hohndel), Матиаса Эттрича (Matthias Ettrich) и всех других хакеров – от борцов с ядром до скромных составителей документации, число которых слишком велико, чтобы всех их здесь перечислить.

Особая благодарность за вклад в проект документирования Linux, техническое рецензирование книги или просто дружескую поддержку Филу Хьюзу (Phil Hughes), Мелинде МакБрайд (Melinda McBride), Биллу Хану (Bill Hahn), Дэну Ирвингу (Dan Irving), Майклу Джонстону (Michael Johnston), Джоэлу Гольдбергеру (Joel Goldberger), Майклу К. Джонсону (Michael K. Johnson), Адаму Рихтеру (Adam Richter), Роману Яновски (Roman Yanovsky), Джону Мэджиду (Jon Magid), Эрику Троэну (Erik Troan), Ларсу Вирцениусу (Lars Wirzenius), Олафу Кирху (Olaf Kirch), Грегу Хенкинсу (Greg Hankins), Алану Сондхейму (Alan Sondheim), Джону Дэвиду (Jon David), Анне Кларк (Anna Clark), Адаму Гудмену (Adam Goodman), Ли Гомесу (Lee Gomes), Робу Уокеру (Rob Walker), Робу Мальде (Rob Malda), Джеффу Бэйтсу (Jeff Bates) и Волькеру Лендеке (Volker Lendecke).

За работу над третьим изданием мы благодарны Филу Хьюзу (Phil Hughes), Роберту Дж. Часселу (Robert J. Chassell), Тони Капеллини (Tony Cappellini), Крейгу Смоллу (Craig Small), Нату Макаревичу (Nat Makarevitch), Крису Дэвису (Chris Davis), Чаку Торопеку (Chuck Toporek), Фредерику Хонгфенгу (Frederic HongFeng) и Дэвиду Пранате (David Pranata) за многочисленные комментарии и исправления. Особое впечатление на нас произвела помощь целой команды разработчиков и пользователей Debian, которую нам предоставили Оссама Отмэн (Ossama Othman) и Джулиан Т. Дж. Мидгли (Julian T. J. Midgley). Джулиан организовал хранение комментариев в системе контроля версий (CVS). Вместе с ним книгу изучали Крис Лоуренс (Chris Lawrence), Роберт Дж. Чассел (Robert J. Chassell), Керк Хиллиард (Kirk Hilliard) и Стефен Зандер (Stephen Zander).

За рецензирование четвертого издания мы благодарны Дэвиду Колье-Брауну (David Collier@Brown), Оливеру Флимму (Oliver Flimm), Филу Хьюзу (Phil Hughes), Крису Лоуренсу (Chris Lawrence), Ричу Пейну (Rich Payne), Крейгу Смоллу (Craig Small), Джеффу Трантеру (Jeff Tranter) и Аарону Веберу (Aaron Weber).

За работу над пятым изданием мы выражаем свою признательность Бену Хайду (Ben Hyde), Чериди Джолли (Cheridy Jollie), Крису Лоуренсу (Chris Lawrence), Эллен Сивер (Ellen Siever) и Джеффу Трантеру (Jeff Tranter).

Калле выражает свою признательность Валерику Ватафу из румынского города Бузау (Buzau), за бесценную помощь при работе над главой, посвященной LAMP. Он также хотел бы поблагодарить своих коллег из компании Klarälvdalens Datakonsult AB – Михаэля Бойера де ла Жироде (Michael Boyer de la Giroday), Таню Далхаймер (Tanja Dalheimer), Стефена Хансена (Stefen Hansen), Джеспера Педерсена (Jesper Pedersen), Лутца Роговски (Lutz Rogowski), Карла-Хайнца Циммера (Kark-Heinz Zimmer), Тобиаса Ларссона (Tobias Larsson), Ромайна Покрживку (Romain Pokrzywka), Дэвида Форэ (David Faure), Марка Муца (Marc Mutz) и Тилла Адама (Till Adam) – за конструктивные комментарии к рукописи и общее «стимулирование Linux-мышления».

I

Удобство и производительность Linux

Эта часть книги служит введением в Linux. Она даст тот минимум знаний, который необходим обычному пользователю для повседневной деятельности: переписка по электронной почте, веб-навигация, игры, просмотр видеофильмов и тому подобное.

Главу 2 стоит прочитать, даже если вы планируете устанавливать Linux из простого в обращении дистрибутива. Здесь рассматриваются такие основополагающие понятия, как выделение дискового пространства для различных частей системы, и говорится о том, что каждая установка операционной системы требует некоторого предварительного планирования.

Подавляющее большинство инсталляций Linux проходит без особых проблем, и в результате пользователи получают в свое распоряжение все функциональные возможности, обсуждаемые в этой части. Если вам придется столкнуться с неприятностями, преодолеть их вам поможет расширенный материал из других частей книги, а также документация в электронном виде и другие, более специализированные источники.

1

Введение в Linux

Добро пожаловать в «Запускаем Linux» пятой версии! Когда мы работали над первым изданием этой книги, операционная система Linux только-только появилась на сцене. Наша задача казалась нам достаточно простой: помочь читателю в освоении основ новой операционной системы, чтобы он научился решать фиксированный, небольшой по объему круг задач. Немногие тогда могли предположить, что со временем Linux станет одной из лучших операционных систем, обеспечивающей поддержку огромного числа программных и аппаратных продуктов, производимых на планете. Кто знал, что число пользователей Linux вырастет от 30 000 человек в 1995 году до сотен миллионов всего лишь за 10 лет? Операционная система Linux нашла применение в самых разных областях, даже в космосе и под водой.

Случайному наблюдателю Linux представляется типичной настольной системой, собранной из тех же строительных блоков, что и любая другая система для персональных компьютеров IBM PC. Многие пользуются Linux для путешествий по Интернету, чтобы получить или отправить электронную почту, послушать музыку или посмотреть фильм, пообщаться с друзьями или коллегами. Студенты и офисные работники создают документы в текстовых процессорах, решают разнообразные задачи с помощью электронных таблиц и создают презентации.

Та же самая операционная система Linux управляет группами сонаров в ядерных подводных лодках, индексирует документы в Интернете, объединяет огромные банки данных на предприятиях, управляет работой почти 70% всех веб-сайтов в мире, записывает телевизионные программы, работает в сотовых телефонах и управляет сетевыми шлюзами, что дает нам возможность общаться с друзьями и близкими, которые могут находиться в любой точке земного шара. Linux работает даже на международной космической станции и на шаттлах, которые доставляют туда астронавтов. Она защищает вас от спама и компьютерных вирусов на огромном числе маршрутизаторов и серверов.

Вы можете обрести преимущества, которые дает эта система, установив ее у себя дома, в школе или в офисе. Мало того что с ее помощью вы по-прежнему сможете решать свои повседневные задачи, вы вдобавок сможете разобраться с тем, как писать запросы к базе данных, как администрировать веб-сервер, как фильтровать электронную почту, отсеивая спам и вирусы, как автоматизировать свой

труд с помощью языков сценариев, как получить доступ к веб-службам и участвовать с помощью компьютера в самых современных видах деятельности.

Как всего этого добиться с помощью Linux? Дистрибутивы Linux заключают в себе огромное количество самых разнообразных технологических достижений, особенно это относится к поддержке новейшего аппаратного обеспечения. Разработчики обладают возможностью доступа ко всему программному коду, который образует единую операционную систему. Многие наблюдатели рассматривают Linux как крупнейший проект по разработке программного обеспечения в истории человечества. Однако на самом деле разработчикам даже не нужно знать о существовании друг друга. Если кто-то захочет написать свою программу, все, что ему нужно сделать, – это загрузить программный код Linux из Интернета или посетить один из сайтов с документацией. Если подсчитать всех разработчиков, сделавших вклад в развитие Linux, их число превысит сотни тысяч.

Разработчики Linux и свободного программного обеспечения принадлежат к самым разным профессиональным слоям. Крупные производители компьютерной техники, такие как IBM, HP, Novell, Red Hat, Sun, Dell и другие, содержат специальные подразделения, занимающиеся развитием Linux. Университеты всего земного шара спонсируют проекты и фонды, которые делают свой вклад в Linux. Министерство обороны США, NASA и Агентство национальной безопасности выделяют солидные суммы на развитие многочисленных подсистем Linux. Многие развивающиеся страны, такие как Китай, Бразилия, Малайзия, Южная Африка и Вьетнам (упомянем лишь некоторые из них), приняли Linux в качестве основной операционной системы. Индустриально развитые страны, например Германия, Австралия, Япония, Великобритания и другие, также явно обозначили свои предпочтения. Но среди этих гигантов живут множество людей, таких как вы или мы, которые также сделали свой маленький вклад в развитие Linux.

В 90-х годах прошлого столетия операционная система Linux вызвала гораздо больший резонанс, нежели любые другие разработки, начиная со времен изобретения микропроцессора. Linux омолодила умирающий сектор компьютерных технологий после кризиса интернет-компаний весной 2001 года. Ныне Linux превзошла самые смелые ожидания даже наиболее информированных наблюдателей, включая и авторов этой книги.

Изначально Linux развивалась благодаря бескорыстной преданности своих пользователей. Специалистам в области интернет-технологий, круг интересов которых включал в себя серверную сторону Интернета, необходимо было изучать операционные системы, под управлением которых работали веб-сайты, службы доменных имен, серверы электронной почты и с помощью которых можно было предоставлять интернет-услуги. Традиционные производители программного обеспечения устанавливали на свои программные продукты такие цены, которые были не по карману тем, кто только начинал осваивать работу веб-мастера. Поэтому многим появление Linux представлялось как удача, потому что они могли совершенно бесплатно загрузить ее из Интернета и получить основные навыки, необходимые для того, чтобы стать настоящим веб-мастером или системным администратором, используя при этом достаточно недорогую аппаратуру.

В узкотехническом понимании Linux представляет собой лишь ядро операционной системы, обеспечивающее базовые службы планирования процессов, виртуальной памяти, управления файлами и обслуживание периферийных устройств,

таких как жесткие диски, приводы DVD, принтеры, терминалы и т. п. Другие операционные системы, способные обеспечивать работу служб Интернета, также принадлежащие семейству UNIX, стали доступны для купли-продажи на коммерческой основе только после разрыва AT&T и Bell Operating Systems.

Чтобы обойти возможные юридические проблемы, связанные с использованием программного кода AT&T UNIX, Фондом свободного программного обеспечения (Free Software Foundation, FSF) было разработано множество приложений, которые реализовали большую часть функций базовой системы UNIX, используя при этом полностью оригинальный программный код FSF и вытеснив программный код, созданный в стенах Bell Labs. Эта коллекция программного обеспечения получила название GNU. Однако для создания полноценной операционной системы необходимо было ядро. Несмотря на то, что работа Фонда в этом направлении застопорилась, совершенно неожиданно операционная система появилась на свет благодаря усилиям студента Хельсинкского университета – Линуса Торвальдса (Linus Torvalds).

Все, кто использует термин «Linux», подразумевают под ним операционную систему, включающую в себя и ядро, и большое число прикладных программ: полноценную среду разработки и рабочее окружение – компиляторы, редакторы, графические интерфейсы, текстовые процессоры, игры и многое другое. Сторонники FSF предложили назвать эту обширную коллекцию программного обеспечения «GNU/Linux».

Об этой книге

Издание представляет собой обзор и руководство по операционной системе Linux. Мы постарались представить достаточный объем общей и интересной информации по ряду тем, чтобы удовлетворить как новичков, так и искушенных пользователей. Книга содержит достаточно полные сведения, необходимые для установки Linux и получения от нее максимальной отдачи. Вместо того чтобы говорить о многочисленных непостоянных технических деталях, имеющих тенденцию к изменению ввиду быстрого развития системы, мы дадим сведения, которых будет вполне достаточно, чтобы преодолеть первые сложности работы с наиболее распространенными дистрибутивами, а также обеспечить начальную подготовку, если в будущем планируется перейти к исследованию более сложных тем, таких как веб-службы, высокопроизводительные вычисления и т. п.

Книга предназначена для тех, кто действительно стремится использовать всю мощь этой операционной системы. Вместо того чтобы дать вам минимум информации, мы поможем вам увидеть, как работают различные части системы, благодаря чему вы сможете самостоятельно исследовать и настраивать систему и устранять возникающие проблемы. Установка и эксплуатация Linux не так сложна, как может показаться на первый взгляд. Однако, как и в случае с любой другой коммерческой операционной системой, здесь так же присутствуют элементы черной магии, и данная книга будет служить вам надежным помощником, если вы планируете использовать Linux не только как настольную систему, но и в качестве сервера сети.

В этой книге будут рассмотрены следующие темы:

- Архитектура и философия операционной системы Linux и те преимущества, которые она может вам дать.
- Сведения о том, что необходимо сделать, чтобы запустить Linux на разных аппаратных платформах, а также порядок настройки системы в зависимости от ее предназначения (например, Linux может использоваться в качестве настольной системы, сервера сети, сервера баз данных или сервера приложений).
- Где приобрести и как установить Linux. В основном мы будем рассказывать о таких дистрибутивах, как Red Hat, SUSE и Debian, но представленных сведений должно быть достаточно для установки дистрибутивов любого другого производителя.
- Введение в основы философии операционных систем UNIX/Linux для новичков, включая обзор наиболее важных команд и понятий.
- Работа с мощными пакетами офисных приложений, в том числе предназначенными для создания графических изображений и ведения бухгалтерского учета.
- Поддержка и обслуживание операционной системы Linux, включая системное администрирование, обновление и устранение неполадок.
- Расширение базовой системы Linux и окружения рабочего стола с помощью дополнительных инструментальных средств для тех, кто склонен к изобретательству.
- Использование Linux для телекоммуникаций и сетевых взаимодействий, включая основы настройки TCP/IP, PPP для модемных соединений с Интернетом, настройку ISDN и ADSL, организацию электронной почты, доступ к телеконференциям и к Web. Мы даже покажем, как настроить систему Linux для работы в качестве веб-сервера.
- Linux для досуга: аудио, видео и игры.

Нам бы хотелось рассказать вам о тысячах вещей, которые можно делать с Linux, но, к сожалению, для этого понадобилась бы книга размером с полный *Оксфордский словарь английского языка*, с которой не справился бы никто (включая несчастных авторов). Вместо этого мы постарались рассказать о наиболее ярких и интересных сторонах системы и сообщить, как можно получить от нее максимум отдачи.

Хотя значительная часть изложенного в книге материала не является сугубо технической, читателю желательно иметь некоторый опыт работы с командной строкой и редактирования простых текстовых файлов. Для тех, кто не обладает таким опытом, мы включили в главу 4 краткое учебное руководство. Во второй части книги описываются приемы администрирования системы, что безусловно пригодится даже опытным пользователям при использовании Linux в качестве сервера.

Если вы новичок в Linux и вам необходимы более подробные сведения о системе, то можно порекомендовать обратиться к какому-либо руководству по основам работы в командной строке. Мы не будем долго задерживаться на основах, предпочитая скорее перейти к более интересным компонентам системы. Большинству читателей будет достаточно этой книги, чтобы приступить к работе с Linux, но многим из вас может потребоваться дополнительная информации по специа-

лизированным темам. Список источников дополнительной информации вы найдете в приложении А.

Кто использует Linux?

Разработчики приложений, системные администраторы, интернет-провайдеры, специалисты по ядру операционных систем, студенты, люди творческих профессий – это лишь некоторые группы людей, для которых Linux обладает особой привлекательностью.

Программисты все чаще используют Linux из соображений стоимости – можно создать полноценную программную среду совершенно бесплатно и пользоваться ею на недорогом персональном компьютере, к тому же Linux представляет собой прекрасную платформу для разработки программного обеспечения, обладающего высокой степенью переносимости. В дополнение к оригинальным инструментальным средствам, разработанным под эгидой FSF, в Linux имеется возможность использовать большое число средств разработки, которые появились за последние три года, как, например, Eclipse (*http://eclipse.org*). Среда разработки Eclipse представляет собой феноменальное явление: результат творческих усилий сообщества open source и тесного сотрудничества между сообществом и крупной компанией (изначально Eclipse была разработана и выпущена компанией IBM). Именно сообщество open source сосредоточило свои усилия на получении расширяемых средств разработки и программной среды, служащей основой для сборки приложений.

Инструментальные средства и платформы, входящие в состав Eclipse, охватывают весь жизненный цикл разработки программного обеспечения, включая поддержку моделирования, средства разработки на языках Java™, C/C++ и других, тестирование и оптимизацию, средства описания бизнес-логики, приложения расширенных клиентов и средства разработки для встраиваемых систем. Платформа Eclipse поддерживается, расширяется и дополняется крупными производителями, исследовательскими группами, университетами и отдельными разработчиками.

Сетевые возможности являются едва ли не самой сильной стороной Linux. Ее с удовольствием восприняли специалисты, которым по долгу службы приходится иметь дело с большими сетями. Они по достоинству оценили простоту администрирования, высокую производительность и низкую стоимость. Многие интернет-сайты используют Linux в качестве платформы для развертывания крупных веб-серверов, приложений электронной торговли, поисковых машин и т. п. Linux поддерживает широко распространенные сетевые стандарты, такие как NFS (Network File System – сетевая файловая система), NIS (Network Information Service – сетевая информационная служба) и другие известные системы, используемые в деловом мире, например систему совместного использования файлов CIFS (Common Internet File System – общая межсетевая файловая система) и протокол LDAP (Lightweight Directory Access Protocol – упрощенный протокол доступа к каталогам). Все это позволяет достаточно просто организовать совместный доступ к файлам, поддерживать регистрацию на удаленных машинах и запускать приложения на других компьютерах. Пакет программного обеспечения под названием Samba позволяет операционной системе Linux выступать в ка-

честве сервера в среде Active Directory. Многие убедились, что сочетание Linux и Samba оказалось более производительным (и более дешевым), чем использование Windows Server 2003. Фактически, учитывая легкость, с которой Linux поддерживает такие сетевые службы, как DHCP, доменную систему имен, систему безопасности на основе Kerberos, маршрутизацию, сложно представить себе сетевую задачу уровня предприятия, которая оказалась бы не по силам этой операционной системе.

Очень часто Linux используется для поддержки крупных промышленных приложений, включая веб-серверы, базы данных, приложения B2B (business-to-business) и сайты для электронной торговли. Большое количество предприятий смогли убедиться на собственном опыте в том, что Linux может служить недорогой, эффективной и устойчивой системой для поддержки самых ответственных приложений.

Вот лишь один пример из множества: компания Cendant Travel Distribution Services перенесла свое приложение Fares на операционную систему Linux Enterprise Server, работающую на аппаратной платформе xSeries и BladeCenter, выпускаемой компанией IBM. В результате общие расходы на поддержку снизились на 90%, уровень доступности повысился до 99,999% при уровне загруженности от 300 до 400 транзакций в секунду.

То обстоятельство, что Linux легко перестраивается для решения конкретных задач вплоть до модификации ядра, делает эту систему очень привлекательной для компаний, которым требуется контроль работы внутренних механизмов системы. Linux поддерживает RAID (механизм, позволяющий работать с массивом жестких дисков как с одним логическим устройством), что позволяет существенно повысить надежность хранения данных. Такое повышение степени надежности совместно с невысокой стоимостью позволяют организовать хранение крупных банков данных целых 30 лет.

Сочетание Linux, веб-сервера Apache, системы управления базами данных MySQL и языка сценариев PHP получило такое широкое распространение, что у него появилась своя собственная аббревиатура – LAMP. Более подробно о LAMP будет говориться в главе 25.

Специалисты по ядру первыми признали Linux, фактически именно они помогли Линусу Торвальдсу создать Linux и до сих пор образуют весьма сплоченное сообщество. Списки почтовой рассылки по ядру Linux отличаются весьма высокой активностью, и если вы хотите оставаться в курсе последних достижений в области проектирования операционных систем, желательно было бы подписаться на них. Linux станет для вас лучшим выбором, если вам нужно отрегулировать алгоритм замены страниц, поэкспериментировать с сетевыми протоколами или оптимизировать работу кэша. На примере Linux можно изучать принципы проектирования операционных систем, и многие университеты используют эту операционную систему в своих специализированных курсах по операционным системам.

Наконец, Linux превратилась в отличную мультимедийную платформу, которая поддерживает огромное количество аппаратных средств, включая большинство современных звуковых карт и видеокарт. Некоторые программные средства, такие как MESA 3D (свободная реализация OpenGL), были перенесены на Linux. Введение в OpenGL будет представлено в разделе «Введение в программирование

с использованием OpenGL» главы 21. GIMP (свободно распространяемый пакет, сходный по своим функциональным возможностям с Adobe Photoshop) изначально разрабатывался под Linux, а ныне он стал одним из любимых инструментов создания графики для многих художников. Многие кинокомпании постоянно используют Linux для создания сложных спецэффектов. Например, при съемках фильмов «Титаник» и «Матрица» для создания сложных спецэффектов использовались рендер-фермы из Linux-машин.

Linux-системы добрались до широт Северного Ледовитого океана, осуществляя передачу и анализ данных на исследовательских океанографических суднах. Операционная система Linux используется на исследовательских станциях в Антарктике, а крупные кластеры, построенные на основе Linux, используются во многих исследовательских учреждениях для моделирования сложных физических явлений типа образования звезд или землетрясений и в лабораториях министерства энергетики, ведущих поиск новых источников энергии. На более приземленном уровне Linux используется в больницах для ведения историй болезней. В Министерстве юстиции США Linux используется для управления всей инфраструктурой, начиная от управления судебными делами и заканчивая бухгалтерским учетом. Многие финансовые учреждения используют Linux для совершения электронных сделок в режиме реального времени. Система Linux сумела принять на себя роль, которую обычно играла операционная система UNIX, как самая надежная.

Характеристики системы

Linux поддерживает большинство функций, присутствующих в других реализациях UNIX и Windows. С дополнениями, которые несет в себе Power Architecture, распространяемая компанией IBM, Linux в состоянии обеспечить такой уровень поддержки аппаратного обеспечения, который доступен только в очень дорогих ЭВМ. Кроме того, последние версии ядра включают в себя поддержку системы расширенной безопасности (Security Enhanced Linux, SELinux), предоставленной Национальным агентством безопасности США (National Security Agency, *http://www.nsa.gov/selinux*). Система SELinux реализует самую безопасную на сегодняшний день вычислительную среду.

Идут работы по добавлению механизма виртуализации в ядро Linux. Механизм XEN (*http://sourceforge.net/projects/xen*) позволяет запустить в одной системе Linux несколько виртуальных машин, каждая из которых может работать под управлением своей собственной операционной системы. Это позволит предприятиям избежать раздувания парка серверов и повысить эффективность использования центрального процессора.

Основные характеристики

Этот раздел представляет собой краткий обзор основных функциональных возможностей операционной системы Linux.

Linux – это многозадачная и многопользовательская операционная система (как и все остальные версии UNIX). Это означает, что одновременно на одной машине могут работать несколько пользователей и выполняться несколько программ. Кроме того, Linux поддерживает многопроцессорные системы (например, мате-

ринские платы с двумя процессорами Pentium), которые могут насчитывать до 32 процессоров[1], что очень ценно для высокопроизводительных серверов и научных приложений.

Операционная система Linux совместима с рядом стандартов UNIX (насколько вообще можно говорить о стандартизации UNIX) на уровне исходных текстов, включая такие стандарты, как IEEE POSIX.1, System V и BSD. Она создавалась с учетом требований, предъявляемых к уровню переносимости исходного программного кода, поэтому вы найдете в Linux функции, присутствующие во многих реализациях UNIX. Большинство бесплатных программ для UNIX, распространяемых через Интернет или иным образом, могут быть скомпилированы в Linux без необходимости внесения дополнительных изменений.

При наличии опыта работы в UNIX вас могут заинтересовать другие внутренние возможности Linux, включая механизм управления заданиями в стандарте POSIX (этот механизм используется в таких командных оболочках, как C shell, *csh* и *bash*), псевдотерминалы (устройства *pty*) и поддержку национальных или специализированных клавиатур с помощью динамически загружаемых драйверов. Кроме того, Linux поддерживает *виртуальные консоли* (*virtual consoles*), которые позволяют в текстовом режиме системной консоли переключаться между разными сеансами регистрации. Тем, кто пользовался программой *screen*, реализация виртуальной консоли в Linux покажется знакомой (хотя почти все предпочитают графическую среду).

Linux благополучно соседствует на одной машине с другими системами, такими как Windows 95/98, Windows NT/2000/XP, Mac OS и другими UNIX-подобными операционными системами, как, например, семейство операционных систем BSD. Начальные загрузчики Linux (LILO) и GRand Unified Bootloader (GRUB) позволяют выбирать, какая операционная система должна загружаться после включения компьютера, при этом Linux совместима и с другими загрузчиками (например, с загрузчиком, который используется в Windows XP).

Linux может работать с широким диапазоном аппаратных архитектур, включая Intel x86 (вся линейка микропроцессоров Pentium), Itanium, SPARC/UltraSPARC, ARM 64 («Hammer»), ARM, PA-RISC, Alpha, PowerPC, MIPS, m68k и мейнфреймы IBM 390 и zSeries. Кроме того, Linux была перенесена на ряд аппаратных архитектур для различных устройств типа PDA, в том числе PalmPilot и Compaq iPaq. С другой стороны, Linux рассматривается в качестве ОС для высокопроизводительных систем. Компания Hewlett-Packard выпустила суперкомпьютер с Linux в качестве основной операционной системы. Кроме того, существует большое число других масштабируемых кластеров – суперкомпьютеров, построенных на базе массивов из персональных компьютеров, работающих под управлением Linux.[2]

---

[1] Для 32-разрядных систем и 64-разрядных архитектур число поддерживаемых процессоров достигает 64. Кроме того, существуют исправления к ядру, которые позволяют увеличить это число до 256.

[2] В этом Linux во многом «обязана» GNU-компилятору *gcc*, который с самого начала был (удачно) выбран в качестве базового для Linux: Linux может работать на архитектурах всех процессоров (перечисленных в тексте), в команды которых «готов» компилировать *gcc*. – *Примеч. науч. ред.*

Linux поддерживает самые разнообразные типы файловых систем для хранения данных. Некоторые файловые системы, например *вторая расширенная файловая система ext2fs*, были специально созданы для Linux. Поддерживаются и другие типы файловых систем UNIX, такие как Minix-1 и Xenix. Реализованы также файловые системы Windows NTFS, VFAT (Windows 95/98) и FAT (MS-DOS), что позволяет непосредственно обращаться к файлам Windows. Кроме того, поддерживаются файловые системы Macintosh, OS/2 и Amiga. Также поддерживается файловая система ISO 9660 CD-ROM, позволяющая читать компакт-диски всех стандартных форматов. Подробнее о файловых системах говорится в главах 2 и 10.

Поддержка сетевых взаимодействий является одной из наиболее сильных сторон Linux как в отношении поддерживаемых функций, так и в отношении производительности. Linux обеспечивает полную реализацию стека протоколов TCP/IP. Она обеспечивает поддержку для многих распространенных сетевых карт Ethernet, протоколов PPP и SLIP (обеспечивают доступ к сетям TCP/IP через последовательный порт и модем), PLIP (Parallel Line Internet Protocol – интернет-протокол через параллельный порт) и ADSL. Дополнительно в Linux имеется поддержка современных протоколов IPv6 и множества других, таких как DHCP, Appletalk, IRDA, DECnet и даже AX.25, предназначенного для обмена пакетами в радиосетях. Поддерживается полный набор клиентов и служб TCP/IP, таких как FTP, Telnet, NNTP и Simple Mail Transfer Protocol (SMTP). Протоколы Sun RPC дают возможность использования NFS (сетевая файловая система) и NIS (сетевая информационная служба), а поддержка протоколов Microsoft позволяет организовать взаимодействия в сетях Microsoft с доменной организацией. Ядро Linux содержит полную поддержку сетевых брандмауэров, которые фильтруют сетевые пакеты, препятствуя, например, несанкционированному доступу к интрасетям.

Широко распространено мнение, что Linux превосходит другие операционные системы в производительности работы сети. Подробнее о сетевых взаимодействиях мы поговорим в главе 13 и в четвертой части книги.

Ядро

Ядро является самой сущностью операционной системы. Это программный код, обеспечивающий взаимодействие пользовательских программ с аппаратурой компьютера, распределение времени между процессами, благодаря которому достигается многозадачность, и многие другие возможности системы. Ядро – это не отдельный процесс, работающий в системе. Его можно представить себе как набор постоянно находящихся в памяти подпрограмм, доступных всем остальным процессам. К подпрограммам ядра можно обращаться разными способами. Самый простой – это обращение к *системному вызову*, который является по сути обычной функцией, заставляющей ядро выполнить некоторый программный код в интересах процесса. Например, системный вызов *read* осуществляет чтение данных из файла. С точки зрения программиста *read* выглядит как обычная функция языка программирования C, но в действительности программный код системного вызова *read* находится внутри ядра.

Тип архитектуры ядра Linux известен как *монолитный*. Это означает, что все драйверы устройств являются частью ядра. В некоторых операционных системах реализована архитектура *микроядра*, когда драйверы устройств и другие компоненты (файловые системы и программный код, управляющий памятью)

не являются частью ядра, а выступают как независимые службы или обычные пользовательские приложения. Обе архитектуры имеют свои достоинства и недостатки. Монолитная архитектура более распространена в реализациях UNIX и применяется в таких классических системах, как System V и BSD. В Linux все же есть поддержка загружаемых драйверов устройств (т. е. таких, которые могут загружаться и выгружаться из памяти по команде пользователя).[1] Эта тема рассматривается в главе 18.

Ядро Linux для платформ Intel было создано с учетом использования специальных возможностей защищенного режима процессоров Intel x86 (начиная с 80386 и далее вплоть до Pentium 4). В частности, Linux использует парадигму управления памятью в защищенном режиме с помощью дескрипторов и другие передовые возможности этих процессоров. Каждый, кто знаком с программированием в защищенном режиме процессора x86, знает, что этот чип проектировался для многозадачных систем, таких как UNIX (x86 фактически возник под влиянием Multics). Linux использует эти функции.

Как и большинство современных операционных систем, Linux является многопроцессорной операционной системой: она поддерживает архитектуры с несколькими процессорами на материнской плате. Это позволяет различным программам работать одновременно (или параллельно) на разных процессорах. Linux также поддерживает *потоки* – распространенную технологию программирования, позволяющую одной программе создать несколько потоков управления, совместно использующих данные в памяти. Linux поддерживает несколько программных пакетов уровня ядра и уровня пользователя, осуществляющих поддержку многопоточных приложений, а потоки ядра Linux выполняются на нескольких процессорах, используя преимущества настоящей многопроцессорной обработки. Реализация потоков ядра Linux совместима со стандартом POSIX 1003.1c.[2]

Ядро Linux поддерживает загрузку страниц исполняемых файлов по запросу. Это означает, что в память с диска считываются только те сегменты программы, которые фактически используются. Кроме того, при одновременном выполнении нескольких экземпляров программы в памяти находится лишь один ее экземпляр. Исполняемые модули работают с динамически связываемыми библиотеками совместного использования. Это означает, что исполняемые модули совместно используют код библиотеки, находящейся в единственном библиотечном файле, расположенном на диске. В результате исполняемые файлы занимают значительно меньшее дисковое пространство. Кроме того, одновременно в памяти находится лишь один экземпляр кода библиотеки, что сокращает общий объем используемой памяти. Можно использовать и статически связываемые библиотеки, если есть стремление к созданию «законченных» исполняемых модулей, которым не нужно наличие совместно используемых библиотек. Динамическое связывание библиотек на этапе исполнения позволяет программистам заменять библиотечные модули своими собственными.

---

[1] Эта техника появилась в Linux относительно поздно (ядро 2.2 или чуть ранее) и известна как «подгружаемые модули ядра». – *Примеч. науч. ред.*

[2] До ядра 2.6 совместимость потоков Linux со стандартами POSIX 1003.1c была достаточно относительной, и только в настоящее время модель потоков Linux приводится в полное соответствие этим стандартам. – *Примеч. науч. ред.*

Для более эффективного использования памяти машины в Linux реализована так называемая *виртуальная память* (*virtual memory*) со страничной подкачкой с диска. Это означает, что на диске может быть выделено пространство для свопинга.[1] Когда приложению требуется больше физической памяти, чем фактически есть в системе, неактивные страницы сбрасываются на диск. (*Страница* – это просто единица измерения памяти, выделяемой операционной системой, в большинстве архитектур размер одной страницы составляет 4 Кбайт.) При очередном обращении к таким страницам они считываются с диска обратно в оперативную память. Такой механизм позволяет запускать приложения большего размера и поддерживать одновременно большее число пользователей. Однако свопинг не может служить равноценной заменой физической оперативной памяти: необходимость чтения страниц с диска замедляет работу.

Ядро Linux хранит в памяти фрагменты файлов, к которым недавно осуществлялся доступ, чтобы избежать лишних обращений к диску (относительно медленных). Вся свободная память системы используется для кэширования обращений к диску, поэтому при невысокой загрузке системы доступ к большому количеству файлов может быстро осуществляться через оперативную память. Когда пользовательским приложениям требуется больший объем физической памяти, размер дискового кэша уменьшается. Таким образом, физическая память всегда используется полностью.

Для облегчения отладки ядро Linux создает *дампы памяти* программ, попытавшихся выполнить недопустимую операцию, такую как обращение к недействительному адресу памяти. Дамп памяти, записываемый в файл с именем *core* в рабочем каталоге программы, помогает программисту определить причину возникновения аварийной ситуации. Об использовании дампов памяти при отладке мы поговорим в разделе «Исследование файла дампа памяти» главы 21.

Команды и командные оболочки

Для многих пользователей самой важной утилитой является *командная оболочка* (*shell*). Командная оболочка – это программа, которая принимает команды пользователя и выполняет их. Кроме того, многие оболочки обладают такими возможностями, как *управление заданиями* (что позволяет пользователю одновременно управлять несколькими процессами – не обращайте внимания, что это звучит, как у Оруэлла), перенаправление ввода-вывода и командный язык, используемый для написания *командных сценариев* (*shell scripts*). Командный сценарий – это файл, содержащий программу на языке командной оболочки, аналогичный командному файлу в операционной системе Windows.

В Linux существует много командных оболочек. Важнейшим различием между ними является командный язык. Например, оболочка C shell (*csh*) использует язык, напоминающий язык программирования C. В классической оболочке Борна используется иной командный язык. Выбор оболочки часто зависит от того,

[1] Строго говоря, это не следует называть пространством для свопинга: выгружаются не процессы целиком, а отдельные страницы памяти. В отдельных случаях могут оказаться выгруженными и целые процессы, но это не обязательно. Термин «swap space» происходит из ранних разработок Linux, и формально следовало бы говорить о «paging space» – страничном файле.

какой командный язык она предоставляет. Используемая оболочка определяет в известной мере вашу рабочую среду в Linux.

К какой бы командной оболочке UNIX вы ни привыкли, ее версия, скорее всего, перенесена на Linux. Самой популярной оболочкой является GNU Bourne Again Shell (*bash*) – «еще одна оболочка Борна». В *bash* включены многие развитые возможности, такие как управление заданиями, история команд, автодополнение команд и имен файлов, Emacs-подобный (или при желании *vi*-подобный) интерфейс редактирования командной строки и мощные расширения языка стандартной оболочки Борна. Другой популярной оболочкой является *tcsh* – разновидность C-оболочки с расширенными функциями, аналогичными имеющимся в *bash*. Среди других командных оболочек можно упомянуть оболочку Корна (*ksh*), *ash* от BSD и *rc* – оболочка Plan 9.

Что важно в этих базовых утилитах? Linux предоставляет уникальную возможность перекроить систему соответственно своим потребностям. Например, если вы являетесь единственным пользователем системы и предпочитаете редактор *vi* и оболочку *bash*, нет причин устанавливать другие редакторы или оболочки. Среди фанатиков и пользователей Linux преобладает позиция «сделай сам».

Обработка текстов и текстовые процессоры

Почти каждому пользователю компьютера требуется та или иная система подготовки документов. (Действительно, один из авторов книги почти разучился писать ручкой на бумаге.) В мире PC обычным явлением стали *текстовые процессоры* (*word processing*): текстовые процессоры предполагают возможность редактирования и обработки текста, зачастую в среде WYSIWYG (What-You-See-Is-What-You-Get – что видишь, то и получаешь), а также изготовление печатных экземпляров текста с рисунками, таблицами и прочими украшениями.

Далее в этой книге вы увидите, что Linux поддерживает весьма привлекательный набор инструментальных средств для работы с текстом в среде WYSIWYG. В главе 8 мы рассмотрим OpenOffice (свободно распространяемая версия коммерческого пакета StarOffice, выпущенного компанией Sun Microsystems, когда она купила фирму – разработчика пакета) и KOffice, каждый из которых представляет собой интегрированный пакет офисных приложений, позволяющих работать с текстовыми документами, электронными таблицами и решать прочие офисные задачи. В них поддерживаются не все функциональные возможности, присущие Microsoft Office, но они обладают рядом интересных особенностей, отсутствующих в Microsoft Office. При желании вы можете по-прежнему пользоваться приложениями из Microsoft Office, запуская их под управлением Wine, о которой мы поговорим позже.

Существуют и другие способы создания документов. Время от времени приходится редактировать файлы настроек системы, набирать тексты программ, для чего вполне достаточно обычных средств *обработки текста*. Наиболее популярными инструментами создания подобных документов являются *vi* и Emacs, которые детально будут описаны в главе 19.

В обработке текста могут использоваться различные инструментальные средства форматирования для создания удобочитаемых документов, имеющих привлекательный внешний вид. При таком подходе автор вводит исходный текст доку-

мента на «наборном языке», который описывает, как должен быть отформатирован текст. По окончании ввода исходного текста (на языке набора) пользователь форматирует его с помощью специальной программы, преобразующей исходный текст в формат, пригодный для печати. Этот процесс в известной мере аналогичен программированию на некотором языке, таком как C, и «компиляции» текста в пригодную для печати форму.

Наиболее известным языком форматирования текста является язык разметки HTML, на котором написаны фактически все страницы в World Wide Web. Еще одним, не менее популярным языком разметки является DocBook XML, своего рода отраслевой стандарт набора тегов для разметки технической документации, используемый и в проекте документирования Linux (обсуждаемом далее в этой главе).

Далее, в главе 20 «Обработка текстовых документов», мы поближе познакомимся с некоторыми системами форматирования, такими как TEX (разработанная Дональдом Кнутом (Donald Knuth), известным авторитетом в области информатики) и ее версией LATEX, *groff*, GNU-версией классического инструмента форматирования *troff*, который первоначально был разработан в стенах Bell Labs, Texinfo (расширение TEX, используемое при создании документации к программному обеспечению Фондом свободного программного обеспечения) и DocBook.

Коммерческие приложения

В дополнение к более чем полутора тысячам приложений для Linux, поддерживаемых дистрибьюторами, такими как Debian, Linux получила широкую поддержку со стороны разработчиков коммерческого программного обеспечения. В число этих программ входят системы автоматизации делопроизводства, текстовые процессоры, научные приложения, средства сетевого администрирования, системы планирования уровня предприятия, такие как Oracle Financials и SAP, и крупномасштабные базы данных. Linux стала играть важную роль на рынке коммерческого программного обеспечения, и количество популярных коммерческих приложений для Linux может поразить. Нет никакой возможности рассказать здесь обо всех них, поэтому мы коснемся самых популярных приложений и кратко отметим ряд других.

Oracle, IBM, Sybase, Informix и Interbase выпустили коммерческие системы управления базами данных для Linux. Многие базы данных для Linux продемонстрировали более высокую производительность, чем соответствующие версии для Windows.

Большой популярностью пользуется такая база данных для Linux, как MySQL – бесплатная и простая в эксплуатации СУБД, которую можно получить по адресу *http://www.mysql.com*. Поскольку MySQL легко установить, настроить и использовать, она быстро стала предпочтительным выбором для тех приложений, которые не нуждаются в сложностях различных коммерческих СУБД. Кроме того, несмотря на свою бесплатность, MySQL поддерживается на профессиональном уровне компанией, которая ее разработала, – MySQL AB. Основы работы с MySQL мы опишем в главе 25.

Все же в MySQL отсутствуют некоторые более развитые функциональные возможности, присущие коммерческим базам данных. Некоторые пользователи предпочитают базу данных с открытым исходным программным кодом Post-

gresSQL[1], и Red Hat включает ее в некоторые свои продукты. Но, с другой стороны, и MySQL не стоит на месте: предполагается, что следующая версия будет включать поддержку распределенных баз данных.

Помимо баз данных для Linux существует целый ряд корпоративных приложений. Linux – одна из самых популярных платформ для поддержки предоставления интернет-услуг, поэтому естественно, что для Linux выпущены высокопроизводительные платформы, предназначенные для создания масштабируемых веб-сайтов, включая JBoss, BEA WebLogic и IBM WebSphere. Существуют также коммерческие, высокопроизводительные виртуальные машины Java, поставляемые Sun, IBM и другими компаниями. IBM выпустила популярный сервер обмена сообщениями и веб-приложений Lotus Domino, а также платформу коммуникаций WebSphere MQ (прежнее название – MQSeries).

Ученые, инженеры и математики обнаружат, что в Linux можно работать с рядом популярных коммерческих продуктов, таких как Maple, Mathematica, MATLAB и Simulink. В число других коммерческих приложений для Linux входят мощные системы автоматизированного проектирования, средства управления сетями, брандмауэры и среды разработки программного обеспечения.

Языки программирования и утилиты

Linux представляет собой полноценную среду разработки для операционной системы UNIX, включая все стандартные библиотеки, средства программирования, компиляторы и отладчики, которые существуют в других UNIX-системах. Чаще всего в Linux для компиляции программ используется GNU Compiler Collection (набор компиляторов GNU), или *gcc*. *gcc* может компилировать программы, написанные на языках C, C++, Objective C (еще один объектно-ориентированный диалект языка C), Chill (язык программирования, используемый в основном для телекоммуникаций), FORTRAN и Java.[2] В мире UNIX разработка приложений и системное программирование обычно осуществляются на языке C или C++, и *gcc* представляет собой один из лучших среди имеющихся компиляторов C/C++, который поддерживает массу развитых функций и оптимизаций.

Java – это одновременно и язык программирования, и среда исполнения, которая поддерживает возможность работы самых разнообразных приложений, таких как апплеты для веб-страниц, распределенные сетевые приложения, прило-

---

[1] Если многие СУБД, в том числе MySQL и подавляющее большинство коммерческих, относятся к классу «реляционных», постулаты которых сформированы относительно давно комиссией под руководством Кодда, то PostgreSQL (ее разработчики используют именно такое написание, а не указанное в тексте) позиционируется как «постреляционная». Кроме этого, в Linux (и во всех других POSIX ОС) представлена еще одна интересная и активно развиваемая СУБД: BercleyDB (как и следует из названия, развиваемая университетом Беркли), принадлежащая к иному классу – «объектных» БД. – *Примеч. науч. ред.*

[2] Начиная с линии версий 3.x.x *gcc* поддерживает так же полноценно язык программирования Ada – известную разработку по заказу Министерства обороны США для «надежного программирования» (этот компилятор один из немногих прошел сертификацию, что обязательно для признания в культуре Ada); Ada развивается в рамках открытого проекта GNAT. – *Примеч. науч. ред.*

жения баз данных и т. п. Linux предоставляет полноценную поддержку Java. Несколько компаний и независимых проектов выпустили свои версии Java Development Kit (набор инструментальных средств для разработки на языке Java) для Linux, среди них можно назвать Sun Microsystems, IBM и Blackdown Project (одним из первых выпустивший реализацию Java для Linux). Программы, написанные на языке Java, могут исполняться в любой системе (независимо от аппаратной или программной архитектуры), главное, чтобы система поддерживала виртуальную машину Java. Существует целый ряд JIT-компиляторов Java (just-in-time), и в поставку Java Development Kit для Linux (JDK) входят высокопроизводительные JIT-компиляторы, работающие не хуже, чем компиляторы для Windows или других UNIX-систем.

Некоторые самые интересные и популярные средства разработки на языке Java относятся к свободно распространяемым продуктам. Среди них можно назвать: Eclipse – интегрированная среда разработки, легко расширяемая с помощью дополнительных модулей, JBoss – реализация платформы Java 2 Enterprise Edition (J2EE) и Gluecode – еще одна прикладная платформа, купленная IBM в мае 2005.

gcc может компилировать Java-программы непосредственно в исполняемые модули и частично поддерживает стандартные библиотеки JDK.

Помимо C, C++ и Java в Linux были перенесены многие другие компилируемые и интерпретируемые языки программирования, такие как Smalltalk, FORTRAN, Pascal, LISP, Scheme и Ada. Кроме того, есть ряд ассемблеров для написания программ в машинных кодах. В рамках одного из достаточно крупных проектов open source, спонсируемого компанией Novell, была разработана среда Mono, реализующая поддержку среды Microsoft .NET в операционных системах UNIX и Linux. Возможно, самую важную группу языков программирования для Linux составляют многочисленные языки сценариев, включая Perl (язык сценариев, являющийся вершиной всех языков этого типа[1]), Python (первый из языков сценариев, изначально проектировавшийся как объектно-ориентированный) и Ruby (исключительно объектно-ориентированный язык сценариев, позиционируемый как одно из лучших средств быстрой разработки).

Для отладки программ в Linux используется развитый отладчик *gdb*, позволяющий выполнять программы в пошаговом режиме при поиске ошибок или анализировать дампы памяти при выяснении причин сбоя программы. Утилита профилирования *gprof* дает возможность получать статистику работы программы для выяснения участков кода, на исполнение которых программа тратит больше всего времени. Текстовые редакторы Emacs и *vim* предоставляют интерактивную среду редактирования и компиляции для различных языков программирования. В числе других инструментов, перенесенных в Linux, можно отметить утилиту GNU *make*, используемую для управления процессом сборки больших программ, а также системы управления версиями исходного кода CVS и *Subversion*.

Linux идеально подходит для создания UNIX-приложений. Она обеспечивает современную среду программирования со всеми дополнительными возможностями, и многие профессиональные UNIX-программисты заявляют, что Linux – их

[1] Точнее, прообразом и прародителем, поскольку каждый из сценарных языков, «оттолкнувшись» от Perl, начал активно развиваться в свою сторону и достиг весьма высокой степени развития. – *Примеч. науч. ред.*

любимая ОС для разработки и отладки. Студенты, обучающиеся по разным направлениям информатики, могут использовать Linux для изучения программирования в UNIX и других особенностей системы, например архитектуры ядра. Linux предоставляет доступ не только к полному набору библиотек и утилит, но и к исходным текстам ядра и библиотек. Языкам программирования и инструментальным средствам, имеющимся в Linux, посвящена глава 20.

Система X Window

X Window System служит стандартным графическим интерфейсом для UNIX-систем. Первоначально она была разработана в MIT (Массачусетский технологический институт) в 80-х годах прошлого века с целью дать возможность приложениям выполняться на рабочих станциях разных производителей, работающих под управлением операционной системы UNIX. X – мощная графическая среда, поддерживающая множество разнообразных приложений. Для X написана масса специальных приложений, включая игры, графические утилиты, средства программирования и работы с документацией и т. п.

В отличие от Microsoft Windows, X Window System располагает встроенной поддержкой сетевых приложений.[1] Например, можно запустить приложение X на сервере, и при этом его окна будут передаваться на рабочую станцию по сети. Кроме того, X поддается настройке в весьма широких пределах: можно изменить по своему вкусу почти любую характеристику системы. Можно менять шрифты, цвета, оформление окон и значки в соответствии со своими предпочтениями. Можно создавать клавиатурные комбинации, нажатием которых будут запускаться связанные с ними приложения. В X можно даже эмулировать рабочий стол Windows или Macintosh, если желательно сохранить привычный интерфейс.[2]

X Window System распространяется свободно. Однако ряд разработчиков стали распространять коммерческие расширения первоначального программного обеспечения X. Версия X Window для Linux известна как X.org, которая является адаптированной версией X11R6 (X Window System Version 11, Release 6), свободно распространяемой для UNIX-систем, устанавливаемых на PC, таких как Linux.[3] X.org поддерживает широкий спектр видеоустройств, включая стандартный

---

[1] Если быть совсем точным, то X-протокол, на котором «стоит» система X Window, – сетевой протокол, так что даже на изолированном локальном компьютере графические X-клиенты взаимодействуют с X-сервером (который и выполняет всю графическую прорисовку) как с сетевым сервером, но установленным на том же компьютере. Поэтому нет ничего удивительного в том, что X-система так легко и естественно «расширяется» на сеть. – *Примеч. науч. ред.*

[2] Точнее, это обеспечивается оконным менеджером, о котором авторы упоминают двумя абзацами ниже, а конкретный оконный менеджер пользователь может выбрать по своему вкусу более чем из десятка имеющихся и свободно доступных. – *Примеч. науч. ред.*

[3] Фактически X.org выросла из другой версии X Window System, предназначенной для установки на персональные компьютеры, – XFree86. Политические раздоры, в которые мы не собираемся здесь вникать, привели к расколу проекта на две части – XFree86 и X.org. Большинство современных дистрибутивов распространяются с версией X.org. Впрочем, для вас это едва ли имеет какое-нибудь значение, если, конечно, вы не собираетесь принять участие в разработке X Window System.

VGA и ряд видеоадаптеров с ускорителями. X.org представляет собой полный комплект программного обеспечения X, включающий в себя сам X-сервер, ряд прикладных программ и утилит, программные библиотеки и документацию. X входит в состав практически каждого дистрибутива Linux.

За внешний вид и поведение интерфейса X Window отвечает *менеджер окон* (*window manager*). Эта дружественная программа отвечает за размещение окон, интерфейс пользователя, посредством которого окна могут изменяться в размерах, сворачиваться, перемещаться, изменять оформление рамок и заголовков и т. д.

Дистрибутив X и основные дистрибутивы Linux кроме всего прочего включают в себя программные библиотеки и заголовочные файлы, предназначенные для программистов, желающих создавать X-приложения, а также все стандартные шрифты, растровые изображения и документацию.

В главе 16 мы расскажем о том, как устанавливать и использовать X Window System на Linux-машине.

KDE и GNOME

Хотя X Window System предоставляет гибкую оконную систему, многие пользователи предпочитают иметь в своем распоряжении законченную среду рабочего стола с возможностью настройки внешнего вида всех окон и графических элементов (таких как кнопки и полосы прокрутки), простым интерфейсом пользователя и развитыми возможностями, как, например, перетаскивание мышью данных между приложениями. KDE и GNOME – два отдельных проекта, имеющие целью предоставить развитую среду рабочего стола для Linux. Путем создания мощного комплекта средств разработки, библиотек и приложений, интегрированных в среду рабочего стола, KDE и GNOME стремятся к тому, чтобы открыть новую эру настольных систем Linux. Эти проекты тесно сотрудничают между собой в духе сообщества open source, чтобы приложения, изначально разрабатывавшиеся для одной среды, могли успешно работать и в другой. Обе системы предоставляют богатый графический интерфейс пользователя, менеджер окон, утилиты и приложения, по своим возможностям не уступающие таким системам, как рабочий стол Windows XP.

Благодаря KDE и GNOME даже неопытные и начинающие пользователи будут чувствовать себя в Linux как дома. Большинство дистрибутивов автоматически настраивают одно из этих окружений рабочего стола во время установки, что исключает необходимость сталкиваться с текстовым интерфейсом консоли.

И KDE, и GNOME ставят перед собой цель сделать Linux более дружественной для пользователя, и каждое их этих окружений имеет своих сторонников и приверженцев. В главе 3 мы познакомимся с ними поближе. Как и X[1], оба окружения рабочего стола предоставляют библиотеки с открытыми исходными текстами, что позволяет разрабатывать программы, соответствующие их духу и внешнему виду.

---

[1] Не совсем точно: и KDE, и GNOME работают «над X», то есть используя X. – *Примеч. науч. ред.*

Работа в сети

Linux считается одной из наиболее мощных и надежных сетевых систем в мире, и все больше людей приходят к выводу, что Linux – это прекрасный выбор для сетевого сервера. Linux поддерживает семейство сетевых протоколов TCP/IP, на которых основан весь Интернет, включая IPv6 (новая версия протокола IP следующего поколения) и UUCP (используемый для связи UNIX-машин через последовательные линии). Linux позволяет связаться с любым компьютером в Интернете через Ethernet (в том числе Fast и Gigabit Ethernet), Token Ring, коммутируемые соединения, беспроводные сети, пакетные радиосети, последовательные линии, ADSL, ISDN, ATM, IRDA, Appletalk, IPX (Novell NetWare) и многие другие сетевые технологии. Доступен весь спектр приложений для Интернета, включая веб-броузеры, веб-серверы, FTP, электронную почту, чат, телеконференции, *ssh*, *telnet* и другие.

Большинство пользователей Linux подключаются к Интернету из дома с помощью коммутируемого соединения с интернет-провайдером. Linux поддерживает популярные протоколы PPP и SLIP, используемые большинством провайдеров для доступа к Интернету по коммутируемым линиям. Если у вас есть широкополосный доступ через канал T1, кабельный модем, DSL или другое устройство, то Linux позволит пользоваться и этими технологиями. Можно даже[1] настроить Linux-машину как маршрутизатор и брандмауэр для нужд сети компьютеров, которые будут подключены к Интернету через одно коммутируемое или широкополосное соединение.

В Linux поддерживаются многие веб-броузеры, в том числе Mozilla (отделившийся от Netscape броузер, распространяемый с открытыми исходными текстами), Konqueror (еще один броузер open source, поставляемый в составе KDE) и текстовый броузер Lynx. Даже текстовый редактор Emacs содержит небольшой текстовый веб-броузер.

Под управлением Linux может также работать ряд веб-серверов. Linux сыграла очень важную роль, благодаря которой появился популярный свободно распространяемый веб-сервер Apache. На сегодня Apache под Linux обслуживает больше веб-сайтов, чем какая-либо другая платформа во всем мире. Apache очень легко устанавливается и настраивается, что мы продемонстрируем в главе 22.

В Linux есть широкий выбор программ для электронной почты и телеконференций, таких как MH, Elm, Pine и mutt, а также средства чтения почты и телеконференций, входящие в веб-броузер Mozilla. Многие из них совместимы со стандартными протоколами, например IMAP и POP. Каковы бы ни были ваши требования, Linux можно настроить для отправки и получения любых сообщений электронной почты и телеконференций.

Есть целый ряд других сетевых служб, которые могут работать под Linux. Samba является пакетом, позволяющим Linux-машинам выступать в качестве файл-сервера и сервера печати Windows. NFS позволяет беспрепятственно совместно использовать файлы вместе с другими машинами сети. Благодаря NFS файлы

[1] Не просто «можно», а это один из наиболее часто применяемых способов связи с Интернетом даже для крупных корпоративных сетей, состоящих исключительно из рабочих станций Windows. – *Примеч. науч. ред.*

на удаленных машинах выглядят так же, как если бы они располагались на ваших собственных дисках.[1] FTP позволяет обмениваться файлами между машинами в сети. В число других сетевых возможностей входят основанные на протоколе NNTP системы электронных новостей, например C News и INN; почтовые агенты Sendmail, Postfix и Exim; *ssh*, *telnet* и *rsh*, позволяющие регистрироваться и выполнять команды на других машинах в сети; *finger*, позволяющий получать информацию о других пользователях Интернета. В общем, существует масса различных приложений и протоколов, основанных на TCP/IP.

Если у вас есть опыт работы с приложениями TCP/IP на других UNIX-системах, то Linux не будет для вас новинкой. Эта система предоставляет стандартный интерфейс программирования сокетов, поэтому практически любая программа, использующая TCP/IP, может быть перенесена на Linux. X-сервер Linux также поддерживает TCP/IP, что позволяет отображать на дисплее приложения, выполняемые на других машинах. Сетевое администрирование окажется знакомым для тех, кто пришел из других UNIX-систем, поскольку средства конфигурирования и мониторинга сходны со своими аналогами в BSD.

В главе 13 мы расскажем о настройке семейства протоколов TCP/IP в Linux, в том числе PPP. Мы разберем также вопросы настройки веб-броузеров, веб-серверов и почтовых программ.

Поддержка портативных компьютеров

В Linux есть ряд функций, специфических для лэптопов, таких как поддержка PCMCIA (или «PC Card») и APM, относительно недавно появившегося ACPI, а также поддержка беспроводных сетей, встроенных в лэптопы Centrino. Пакет PCMCIA Tools для Linux содержит драйверы многих устройств PCMCIA, включая модемы, карты Ethernet и адаптеры SCSI. APM позволяет ядру следить за уровнем зарядки аккумулятора и выполнять определенные действия (например, автоматически завершать работу) при снижении его ниже допустимого уровня. Кроме того, эта функция переводит процессор в режим малого энергопотребления, когда он не используется. Такую функцию легко настроить как параметр ядра. Есть ряд утилит для взаимодействия с APM, в том числе *apm*, которая отображает состояние аккумулятора, и *apmd*, которая регистрирует состояние аккумулятора и может генерировать события, связанные с состоянием питания. Эти утилиты присутствуют в большинстве дистрибутивов Linux. Аналогичным целям служит поддержка ACPI, но эта функция появилась сравнительно недавно и обладает более широкими возможностями. С помощью функции ACPI можно даже настроить возможность сохранения на диске текущего состояния компьютера при его выключении. Благодаря этому при последующем включении компьютер продолжит работу точно с того места, когда он был выключен. А программные инструментальные средства, обладающие графическим интерфейсом, как, например *kpowersave*, позволяют управлять всеми этими возможностями, не покидая удобной и дружественной графической среды.

---

[1] Благодаря не только NFS, но в еще большей мере общей концепции файловых систем, принятых в UNIX и, следовательно, в Linux. – *Примеч. науч. ред.*

Взаимодействие с Windows

Существуют различные утилиты для взаимодействия с миром Windows и MS-DOS. Наиболее известным приложением является проект Wine – платформа для запуска приложений, разработанных для Microsoft Windows, в X Window System под Linux. Wine позволяет программам Microsoft Windows выполняться непосредственно под управлением Linux и других операционных систем на платформе Intel. Wine находится в процессе непрерывного развития и в настоящее время позволяет запускать многочисленные[1] программы для Windows, включая настольные приложения и игры. О подробностях этого проекта мы поговорим в главе 28.

Linux обеспечивает прозрачный интерфейс для перемещения файлов между системами Linux и Windows. Под Linux можно смонтировать раздел Windows или гибкий диск и непосредственно обращаться к файлам Windows, как к любым другим. Кроме того, существует пакет *mtools*, который дает возможность непосредственного доступа к гибким дискам, отформатированным в MS-DOS, и пакет *htools*, делающий то же самое в отношении гибких дисков Macintosh.

Еще одним устаревшим приложением из этой области является Linux MS-DOS Emulator, или DOSEMU, позволяющее запускать многие программы для MS-DOS прямо из Linux. Несмотря на то, что программы MS-DOS быстро уходят в небытие, все же существует ряд интересных утилит и игр под MS-DOS, которые хотелось бы запускать в Linux. В DOSEMU можно даже запускать старую программную оболочку Microsoft Windows 3.1.

Linux не предоставляет полной эмуляции среды Windows и MS-DOS, но можно запускать эти ОС на той же машине, где стоит Linux, и выбирать операционную систему для запуска во время начальной загрузки. Многие дистрибутивы во время установки умеют распознавать и сохранять нетронутыми другие операционные системы, которые ранее уже были установлены на компьютере, а благодаря установке начального загрузчика LILO или GRUB позволяют выбирать между Linux, Windows и другими ОС при начальной загрузке. В этой книге мы покажем, как правильно настроить начальный загрузчик LILO на тот случай, если вам понадобится организовать выбор загружаемой операционной системы вручную.

Еще одним популярным решением является запуск на системном уровне виртуальной машины, которая позволяет работать операционным системам Linux и Windows *одновременно*. *Виртуальная машина* – это прикладная программа, которая эмулирует большую часть аппаратных средств машины, заставляя операционную систему поверить в то, что она работает на реальном физическом компьютере. С помощью виртуальной машины можно загрузить Linux, а затем запустить Windows – и на машине будут одновременно выполняться приложения Linux и Windows. Можно сделать наоборот – загрузить Windows, а затем запустить на виртуальной машине Linux. Хотя при работе с виртуальной машиной несколько снижается производительность, часто очень удобно применять такую технологию, чтобы, например, запустить под Linux текстовый процессор, разработанный для Windows. Наибольшее распространение получили такие виртуаль-

[1] Но не все: возможность или невозможность выполнения программы для Windows под Wine зависит от специфики этого конкретного приложения – от того, какие механизмы Windows API оно использует. – *Примеч. науч. ред.*

ные машины, как VMWare (*http://www.vmware.com/*), которая является коммерческим продуктом, и Bochs (*http://bochs.sourceforge.net/*), представляющая собой проект с открытыми исходными текстами. Описание VMWare вы найдете в главе 28.

Наконец, возможность удаленной регистрации, имеющаяся в Linux, позволяет выполнять действия на удаленных системах. Любые два компьютера, на которых работает X Window System (главным образом это операционные системы Linux, BSD и UNIX[1]), позволяют пользователю, находящемуся за одним компьютером, запускать программы на другом, работать с графическим интерфейсом программы, который выводится на его дисплее, и управлять им с помощью своей клавиатуры и мыши. Протокол RDP (эта аббревиатура может расшифровываться и как Remote Desktop Protocol – протокол управления удаленным рабочим столом, и как Remote Display Protocol – протокол управления удаленным дисплеем) позволяет аналогичным образом управлять программами, исполняющимися на удаленных Windows-системах. Клиентская и серверная части Virtual Network Connection (VNC – виртуальное сетевое соединение) выполняют те же самые задачи, причем с еще большей гибкостью, позволяя совместно работать различным операционным системам на разных компьютерах. В разделе «Доступ к удаленному рабочему столу Windows» мы продемонстрируем, как настраиваются эти службы, а в разделе «FreeNX: Linux как сервер удаленного рабочего стола» обсудим систему удаленных взаимодействий FreeNX, которая позволяет организовать прозрачное сетевое взаимодействие, аналогичное X, с огромным преимуществом в скорости.

Другие приложения

Для Linux существует масса приложений, чего и следовало ожидать от операционной системы с таким разнообразным составом пользователей. В настоящее время Linux сосредоточена главным образом на организации персональных рабочих мест в UNIX, но ситуация быстро меняется. Расширяется спектр приложений для деловых и научных целей, появляется все больше коммерческих приложений.

В научном мире Linux бесспорно считается лучшей платформой для экономичного решения вычислительных задач. Для Linux разработано большое число научных программ, включая популярные технические инструменты MATLAB и Mathematica. Есть также большое число бесплатных пакетов, в том числе FELT (Finite Element Analysis Tool – анализ методом конечных элементов), Spice (средство проектирования и анализа электронных схем) и Khoros (система обработки изображений, цифровых сигналов и визуализации). На Linux перенесены многие популярные библиотеки вычислительных алгоритмов, например LAPACK – библиотека методов линейной алгебры. Есть также оптимизированная для Linux версия кода BLAS, на котором основывается LAPACK.

Linux – одна из самых популярных платформ для организации параллельных вычислений с помощью кластеров, которые представляют собой группу недоро-

[1] Для Windows также существует несколько реализаций X-системы от независимых производителей; наиболее известной из них является, по-видимому, Exceed. – *Примеч. науч. ред.*

гих машин, обычно соединенных между собой высокоскоростной (порядка нескольких гигабит в секунду или выше) сетью. Проект NASA Beowulf первым популяризировал идею связать большое число PC, работающих под Linux, в один суперкомпьютер, чтобы осуществлять научные или числовые расчеты. В настоящее время кластеры на базе Linux стали скорее правилом, чем исключением, во многих научных приложениях. Кластеры Linux находят себе применение в самых разнообразных приложениях. Например, поисковая машина Google использует кластер из Linux-машин (согласно утверждениям MIT их число в декабре 2004 года превысило 250 000)!

Как и во всякой операционной системе, в Linux есть свои игры. На Linux переведен ряд популярных коммерческих игр, в том числе Quake, Quake II, Quake III Arena, Doom, SimCity 3000, Descent и многие другие. Большинство популярных игр позволяют играть через Интернет или локальную сеть. Возникают Linux-клоны других коммерческих игр. Есть также классические текстовые подземельные игры, такие как Nethack и Moria; игры типа MUD (многопользовательские «темницы», позволяющие нескольким пользователям участвовать в текстовых приключениях), например DikuMUD и TinyMUD; множество графических игр, таких как *xtetris*, *netrek* и *Xboard* (X11-интерфейс для *gnuchess*).

Меломаны найдут в Linux широкую поддержку различной звуковой аппаратуры и сопутствующих программ, таких как CDplayer (программа, управляющая приводом CD-ROM как обычным CD-плеером, что довольно удивительно), MIDI-секвенсоры и редакторы (позволяющие сочинять музыку для воспроизведения через синтезатор или другой инструмент, обладающий поддержкой MIDI) и звуковые редакторы для оцифрованного звука. В Linux можно прослушивать аудиофайлы самых разных форматов, такие как MP3 и OGG/Vorbis, а с помощью разнообразных программ, входящих в состав некоторых дистрибутивов, становятся доступны для прослушивания даже файлы некоторых коммерческих форматов.

Не можете найти нужную программу? Есть ряд веб-сайтов, на которых находятся обширные каталоги приложений для Linux. Наиболее популярным каталогом Linux-программ является Freshmeat (*http://www.freshmeat.net*); перечень, включающий другие известные каталоги, вы найдете в приложении A. Посмотрите эти сайты, чтобы, по крайней мере, убедиться, как много программ написано для Linux.

Если требуемое не удается найти, всегда можно попытаться перенести на Linux приложение с другой платформы. В крайнем случае можно написать приложение самому. Это в духе Free Software – если хочешь, чтобы что-то было сделано хорошо, сделай это сам! Начинать одному крупный программный проект бывает страшно, но те, кто открывает публике раннюю версию своей программы, обнаруживают, что возникает множество помощников, готовых присоединиться к проекту.

Об авторских правах на Linux

Linux подпадает под действие лицензионного соглашения, которое известно как GNU *General Public License* (Универсальная общественная лицензия GNU), или *GPL*. GPL, которую иногда называют лицензией «copyleft» («права оставлены», как противоположность copyright – «права удержаны»), разработана Free Soft-

ware Foundation (Фонд свободно распространяемого программного обеспечения) для проекта GNU. Она накладывает ряд условий на распространение и модификацию свободного программного обеспечения. В этом смысле слово «свобода» подразумевает свободу распространения, а не только возможность пользоваться бесплатно. GPL всегда была подвержена неправильному толкованию, и мы надеемся, что данный обзор поможет понять рамки и цели GPL, а также ее отношение к Linux. Полный текст GPL можно найти по адресу *http://www.gnu.org/copyleft/gpl.html*.

Первоначально Линус Торвальдс выпустил Linux под более строгой, чем GPL, лицензией, которая допускала свободное распространение и модификацию, но запрещала денежные расчеты за распространение и использование. GPL разрешает продавать и получать прибыль от свободного программного обеспечения, но запрещает каким-либо образом ограничивать права на дальнейшее распространение программного обеспечения.

Краткое описание принципов лицензирования свободного программного обеспечения

Прежде всего, мы хотим разъяснить, что свободное программное обеспечение, охватываемое GPL, *не* подразумевает общественное владение (public domain). Программное обеспечение в общественном владении не имеет авторских прав и находится в общественном владении в буквальном смысле этого слова. Напротив, программное обеспечение, подпадающее под действие GPL, имеет авторские права, принадлежащие одному или нескольким авторам. Это значит, что программное обеспечение защищено обычными международными законами об авторских правах и автор его юридически определен. Тот факт, что программное обеспечение может свободно распространяться, не означает, что оно находится в общественном владении.

Программное обеспечение с лицензией GPL также не является условно бесплатным (shareware). Обычно программное обеспечение с лицензией shareware и авторские права на него принадлежат автору, который требует, чтобы пользователи платили некоторые суммы за пользование полученным программным обеспечением. Напротив, программное обеспечение с лицензией GPL может бесплатно распространяться и использоваться.

GPL разрешает также модификацию свободного программного обеспечения и распространение модифицированных версий. Однако результаты модификации также подпадают под действие GPL. Иными словами, фирма не может взять Linux, модифицировать ее и продавать под ограничительной лицензией. Если какое-либо программное обеспечение является производным от Linux, оно также должно быть охвачено лицензией GPL.

Лица и организации могут распространять программное обеспечение GPL за плату и даже извлекать прибыль из продажи и распространения, однако продавец программного обеспечения GPL не может лишить таких же прав покупателя. Это значит, что если вы где-либо приобрели ПО, распространяемое на условиях GPL, то сами можете распространять его бесплатно или продавать.

Поначалу в этом можно усмотреть противоречие. Как можно с прибылью продавать программное обеспечение, когда GPL позволяет любому приобрести его бес-

платно? Когда фирма собирает вместе много бесплатных программ, помещает их на компакт-диск и распространяет, ей должны быть оплачены расходы на производство и распространение компакт-диска, а кроме того, она может пожелать получить прибыль от продажи программного обеспечения. GPL это разрешает.

Организации, продающие свободное программное обеспечение, должны следовать некоторым ограничениям, установленным GPL. Во-первых, они не могут ограничить права пользователей, приобретающих программное обеспечение. Это значит, что, купив компакт-диск с программным обеспечением GPL, вы можете копировать и распространять его бесплатно или перепродавать самостоятельно. Во-вторых, распространитель должен сделать очевидным для пользователей, что программное обеспечение действительно подпадает под действие GPL. И в-третьих, распространитель должен бесплатно предоставить полный исходный код распространяемого программного обеспечения или указать по требованию покупателя, откуда он может быть загружен.[1] Поэтому каждый покупатель программного обеспечения GPL может модифицировать это программное обеспечение.

Разрешение распространять и продавать свободное программное обеспечение – очень полезная вещь. Не у всех есть доступ к Интернету для бесплатной загрузки такого программного обеспечения, как Linux. GPL позволяет фирмам продавать по разумной цене или поставлять бесплатно программное обеспечение тем, у кого нет возможности загрузить его из Интернета. Например, многие организации продают Linux на гибких дисках, лентах или компакт-дисках по почте и получают от этого прибыль. Разработчикам Linux из этой прибыли, вероятно, ничего не достанется. Таковы отношения между разработчиком и распространителем, устанавливаемые GPL. Иными словами, Линус знал, что фирмы могут начать продавать Linux, и он не получит из их прибыли ни гроша. (Если Линус и не богат, то по крайней мере он знаменит!)

В мире свободного программного обеспечения важны не деньги. Цель свободного программного обеспечения – разработка и распространение фантастических программ и возможность каждого получить их и пользоваться ими. В следующем разделе мы обсудим, как это отражается на разработке Linux.

SCO и другие компании, оспаривающие права на Linux

В марте 2003 года компания с названием SCO, история развития которой шла весьма извилистым путем слияний и разоблачений, связанных с покупкой некоторого количества прав на UNIX, заявила, что Linux содержит исходные тексты, правами на которые обладает SCO, и потому права на Linux также принадлежат SCO. Компания начала судебные разбирательства, возбудив иск против IBM. Это был, по крайней мере, слишком самоуверенный выпад, потому что трудно представить себе компанию, сделавшую бизнес в компьютерной индустрии, которая была бы лучше подготовлена к судебным разбирательствам. Как бы то ни было, но позднее выяснилось, что притязания SCO не ограничились одной только IBM.

---

[1] Хотелось бы акцентировать внимание именно на этом пункте, который является краеугольным камнем и отличает лицензию GPL от других (и их немало) свободных (бесплатных в использовании) лицензий, но прозвучал в тексте как-то «мимоходом»: программное обеспечение под GPL должно быть безусловно доступным в виде исходных текстов! – *Примеч. науч. ред.*

В действительности SCO имела некоторое отношение к Linux. Согласно отчетам в декабре 2003 года SCO даже разослала предупредительные письма более чем 1000 компаний, предлагая выплатить лицензионные отчисления SCO.

К разбирательствам присоединились Red Hat и ряд других компаний. Novell, которая в то время занималась покупкой SUSE и стала верным партнером сообщества Linux, еще больше подогрела интерес к этому делу, предъявив свои собственные права на UNIX. В результате затянувшееся дело превратилось в целую серию судебных процессов, встречных исков, отказов от притязаний, громких заявлений перед общественностью и поливания друг друга грязью.

К моменту написания этих строк конфликт с компанией SCO еще не был урегулирован, но теперь дело уже не выглядит угрожающим. Не многие наблюдатели полагают, что Linux в беде, – скорее SCO оказалась перед угрозой финансового краха. Сеть компаний, отдельных лиц и ключевые организации, поддерживающие Linux, достойно ответили на брошенный им вызов. Некоторые крупные поставщики еще больше усилили поддержку Linux, предложив своим пользователям компенсационные выплаты. Мы надеемся, что в следующем издании нашей книги информации об этом деле будет немного больше.

Наконец Линус Торвальдс и OSDL признали, что старый порядок принятия программного кода без надежной привязки изжил себя. Начиная с мая 2004 года, любой разработчик, представивший программный код для включения в ядро, должен указать контактную информацию и неофициально объявить о своем авторском праве на представленный код. Новая система получилась простой и легкой, но при этом сделала возможным в случае чего обратиться к тем, кто взял на себя ответственность за рассматриваемый код.

Дальнейшие попытки оспаривания авторских прав на Linux маловероятны, но против нее могут использоваться патенты. Однако каждый программист и каждая компания, занимающиеся разработкой программного обеспечения, должны волноваться о программных патентах. Linux и бесплатное программное обеспечение ничуть не опаснее, чем любое другое программное обеспечение. Проекты по свободному программному обеспечению ведутся открыто, и поэтому они вполне могут стать соблазнительной целью для возбуждения новых судебных разбирательств. Единственная цель таких разбирательств состоит в том, чтобы преднамеренно прекратить работу проекта, потому что свободное программное обеспечение не может приносить лицензионных отчислений.

Программное обеспечение с открытыми исходными текстами и философия Linux

У начинающих пользователей Linux часто встречается ряд заблуждений и неверных представлений о системе. Linux является уникальной операционной системой, для эффективного использования которой важно понимать ее философию и архитектуру. В центре философии Linux лежит идея, которую мы называем сейчас программным обеспечением с открытыми исходными текстами (open source).

«Open source» означает программное обеспечение, исходный программный код которого (т. е. внутреннее устройство программы) каждый желающий может

свободно загрузить, изменить и повторно распространить.[1] Программное обеспечение, подпадающее под условия GPL, описанной в предыдущем разделе, укладывается в категорию open source. Никого не должно удивлять, что в ту же категорию попадает масса программного обеспечения, распространяемого на условиях других лицензий, сходных с GPL, но не совпадающих с ней. Например, программное обеспечение, которое можно свободно модифицировать, но к которому при этом не предъявляются такие же строгие требования по последующему распространению, как в GPL, тоже считается open source. В эту категорию попадают самые разные лицензии, в том числе BSD License и Apache Software License.

Модели разработки программного обеспечения с открытыми исходными текстами и свободного программного обеспечения – это явление, начавшееся с Фонда свободного программного обеспечения и ставшее популярным во многом благодаря Linux. Это совершенно новый способ создания программного обеспечения, при котором все стороны разработки, отладки, тестирования и исследования открыты всем, кто проявит к ним интерес. Не полагаясь на какую-то одну фирму в разработке и поддержке программ, модель open source обеспечивает открытое развитие программного кода в сообществе разработчиков и пользователей, движимых желанием *создавать хорошее программное обеспечение*, а не стремлением к прибыли.

Издательство O'Reilly опубликовало две книги «Open Sources 1.0» и «Open Sources 2.0», которые могут служить прекрасным введением в модель разработки open source. Они представляют собой сборники очерков о движении open source, написанных ведущими разработчиками (включая Линуса Торвальдса и Ричарда Столлмана). Еще одна не менее популярная книга по этой же теме – «The Cathedral and the Bazaar», принадлежащая перу Эрика С. Реймонда (Eric S. Raymond).

Пресса уделила большое внимание модели open source, и некоторые называют это явление новой волной в разработке программного обеспечения, которая заставит забыть прежние методы, когда все скрывалось от посторонних глаз под покровом секретности. Со временем станет видно, насколько жизнеспособно это движение, но уже сейчас имеются вполне обнадеживающие признаки того, что это не столь уж несбыточно. Например, Netscape Corporation опубликовала исходные тексты своего веб-броузера как проект open source под названием Mozilla. Другие крупные компании, такие как Sun Microsystems, IBM и Apple, объявили о планах выпуска некоторых своих продуктов как open source в надежде, что в условиях совместного общественного проекта разработки программного обеспечения они достигнут успеха.

---

[1] Еще важнее, чем возможность модификации и улучшения: а) возможность анализа и верификации программного кода для принятия решения о его применении в критически важных отраслях (например, управление реактором электростанции или железнодорожным узлом) и б) возможность гарантировать отсутствие скрытых программных «закладок» для приложений, применяемых в областях, затрагивающих безопасность государства (например, в области военных систем, области противодействия чрезвычайным ситуациям и т. п.). Именно по этим причинам ряд государств (Китай и др.) приняли Linux в качестве «официальных» ОС для применения в государственной сфере, о чем авторы упоминали ранее. – *Примеч. науч. ред.*

Linux находится в центре внимания прессы к явлению open source. Чтобы понять, откуда происходит идеология разработки Linux, есть смысл взглянуть на традиционную модель разработки коммерческого программного обеспечения.

Коммерческие компании стремятся вести разработку на основе строгой политики контроля качества, использования средств контроля исходного программного кода и систем контроля версий, документирования, учета выявленных ошибок и их устранения. Разработчикам не разрешается по своей прихоти добавлять новые функциональные возможности или изменять ключевые участки кода: изменения могут производиться только в ответ на сообщения об ошибке, причем все изменения вводятся в систему контроля версий, чтобы при необходимости их можно было откатить. За каждым разработчиком закреплена одна или несколько частей программного кода, и только этот разработчик может изменять эти части, когда они «взяты для редактирования».

Внутри компании отдел технического контроля на каждом очередном проходе программного обеспечения выполняет наборы строгих тестов (так называемые регрессионные тесты) и составляет отчеты об обнаруженных ошибках. Устранение этих ошибок является обязанностью разработчика. Применяется сложная система статистического анализа для того, чтобы при выпуске очередной версии был устранен определенный заданный процент ошибок и операционная система в целом удовлетворила некоторым выходным критериям.

В целом процесс сопровождения и поддержки программного кода разработчиками коммерческого программного обеспечения очень сложен, и это правильно. Компания должна иметь количественный критерий того, что очередная версия программного обеспечения готова к выпуску. Разработка коммерческой программной системы – очень большая работа, для выполнения которой часто требуются сотни (если не тысячи) программистов, тестеров, составителей документации и административного персонала. Конечно, у каждой компании-разработчика есть свои особенности, но в целом картина такова. Небольшие компании обычно копируют такую организацию в уменьшенном масштабе.

Полную противоположность этой модели разработки представляет Linux, которая была и наверняка останется хакерской операционной системой.[1] Хотя многие проекты open source приняли элементы коммерческой методологии разработки приложений, такие как системы контроля версий и регистрации ошибок, все же неотъемлемой чертой разработки Linux являются сотрудничество и рассредоточение, что в корне отличает эту модель от традиционного подхода.

Не так давно шло много разговоров о методологиях гибкой разработки, таких как экстремальное программирование. Адепты Linux и движения open source часто немного удивляются таким разговорам, так как подобные методологии всегда были центральной идеей разработки программного обеспечения с открытыми исходными текстами.

---

[1] Под словом «хакер» мы подразумеваем программиста, увлеченного своей работой, – человека, который получает массу удовольствия от работы за компьютером и, как правило, делает с его помощью что-либо интересное. Такое толкование термина «хакер» противоречит устоявшемуся представлению о хакере как о человеке, делающем с помощью компьютера что-то нехорошее и противоправное.

Linux разрабатывается главным образом как коллективный проект группы добровольцев из Интернета, живущих в разных концах света. Единой организации, ответственной за разработку системы, не существует. В основном Linux-сообщество общается с помощью различных почтовых списков рассылки и веб-сайтов. В ходе работы над проектом возник ряд соглашений, например, программисты, которые хотят, чтобы их программный код был включен в «официальное» ядро, должны послать его Линусу Торвальдсу. Он проверит код и включит его в ядро (если код ничего не портит и не противоречит общей архитектуре системы, то, скорее всего, он это сделает). По мере развития Linux эта задача стала слишком трудоемкой, чтобы Линус мог справиться с ней один (к тому же он теперь обзавелся детьми), поэтому часть его обязательств взяли на себя другие участники проекта, отвечающие за тестирование кода и включение его в определенные подсистемы ядра, например в сетевую подсистему.

Архитектура самой системы делает ее весьма открытой для добавления новых возможностей и функций. Новая версия ядра обычно выходит раз в несколько недель, а иногда и чаще. Конечно, это очень приблизительная цифра, все зависит от того, сколько обнаружено ошибок, подлежащих исправлению, велик ли поток сообщений от пользователей, тестирующих предварительные версии, и много ли удалось Линусу поспать в течение недели.

Следует сказать, что выход новой версии не означает устранение всех до единой ошибок и решение всех проблем. (В равной степени это относится и к коммерческим программам!) Как только система представляется свободной от критических или часто возникающих ошибок, она признается стабильной и осуществляется выпуск новой версии. Главная задача разработки Linux не в том, чтобы создать совершенный и лишенный ошибок программный код, цель состоит в разработке свободной реализации UNIX. Больше чем для кого бы то ни было Linux предназначается для разработчиков.

Каждый, кто хочет добавить к системе новую функциональную возможность или программу, обычно делает ее доступной в версии «альфа». Это стадия, когда программа тестируется теми отчаянными пользователями, стремящимися устранить проблемы в исходном программном коде. Поскольку Linux-сообщество базируется в основном в Интернете, альфа-версии обычно выкладываются на одном или нескольких веб-сайтах Linux (см. приложение А), а в списки рассылки Linux помещается сообщение о том, как получить и протестировать этот код. Пользователи, загрузившие и протестировавшие программу, могут по электронной почте сообщить автору о полученных результатах, исправленных ошибках и задать ему вопросы.

После устранения первоначальных проблем альфа-версия переходит в стадию «бета», на которой программный код обычно считается стабильным, но не завершенным (т. е. могут быть реализованы не все функции). Иногда программа может сразу перейти в финальную стадию, когда программа считается законченной и пригодной к использованию. Если это код ядра, то по завершении разработчик может попросить Линуса о включении его в стандартное ядро или использовать в качестве необязательной дополнительной функции ядра.

Учтите, это лишь общепринятые соглашения, но не правила. Некоторые разработчики настолько уверены в своих силах, что не выпускают версий в стадии «альфа» или «бета». Такие решения всегда принимает сам разработчик.

Что же происходит с регрессионным тестированием и строгим процессом контроля качества? Вместо него действует правило «выпускай скорее и выпускай чаще». Лучшими контролерами являются сами пользователи, поскольку они испытывают систему во множестве сред и с разнообразными реальными приложениями, что не так просто осуществить в любой штатной группе контроля качества. Одной из самых примечательных сторон такой модели разработки и выпуска является то, что ошибки (и бреши в системе безопасности) порой могут быть выявлены, объявлены и устранены в течение нескольких *часов*, а не дней и недель.

Поразительно, что такая неорганизованная система добровольцев, программирующих и отлаживающих целую UNIX-систему, может вообще что-то произвести. Как показывает практика, это один из наиболее эффективных и целеустремленных программных проектов, существовавших когда-либо. Ядро Linux было целиком написано, что называется, *«с нуля»*, без использования какого-либо коммерческого программного кода. Добровольцами была проделана огромная работа по переносу на Linux всего существующего на свете бесплатного программного обеспечения. Были написаны и перенесены библиотеки, созданы файловые системы, написаны драйверы для многих распространенных аппаратных устройств.

Программное обеспечение Linux обычно выпускается в виде *дистрибутива*, представляющего собой единый пакет программ, в целом образующих систему. Большинству пользователей было бы слишком сложно вручную собрать систему с самого начала, для чего потребовалось бы сначала собрать ядро, а затем добавить утилиты и установить все необходимое программное обеспечение. Вместо этого предлагается целый ряд дистрибутивов, в состав которых входит все необходимое для установки и запуска системы. Опять-таки, стандартного дистрибутива не существует, их много, и у каждого есть свои преимущества и недостатки. Эта книга описывает установку дистрибутивов Red Hat, SUSE и Debian, но она может оказаться полезной при установке любого другого дистрибутива.

Несмотря на то, что Linux является завершенной системой, для установки и запуска полной системы требуется некоторый опыт работы в UNIX. Нет такого дистрибутива Linux, в котором не нашлось бы каких-то ошибок, поэтому после инсталляции может потребоваться вручную исправить небольшие проблемы. Некоторым читателям это обстоятельство может доставить огорчение, но правильнее было бы воспринимать его как «радость Linux», то есть удовольствие, получаемое от возни со своей собственной системой, изучения ее и исправления мелких недочетов. Именно такое отношение отличает энтузиастов Linux от простых пользователей. Linux может быть хобби, увлекательным спортом или стилем жизни. (Так же, как у любителей сноуборда или горного велосипеда, у «линуксоидов» есть свой жаргон и стиль одежды. Если не верите – потолкайтесь на какой-нибудь выставке Linux!) Многие начинающие пользователи Linux пишут о том, с каким удовольствием они осваивают эту новую систему, и считают, что Linux воскрешает очарование, испытанное при первом знакомстве с компьютером.

Советы начинающим пользователям UNIX

Для установки и запуска собственной Linux-системы не требуется большого опыта работы в UNIX. На самом деле многие начинающие пользователи успешно справляются с установкой Linux. Приобрести опыт установки полезно, но уч-

тите, что некоторым процедура установки может показаться весьма сложной и запутанной. Если вам повезет, вы сможете установить Linux и начать в ней работать, вообще не имея опыта работы с UNIX. Однако, как только вы будете готовы заняться более сложными задачами – установкой новых программ, пересборкой ядра и т. п., – знание UNIX станет для вас необходимостью. (Отметим при этом, что многие дистрибутивы Linux так же просто установить и настроить, как Windows 98, и уж конечно проще, чем Windows 2000 или XP.)

К счастью, после установки собственной Linux-системы можно приступать к изучению основ UNIX, которые необходимо знать для выполнения этих задач. В этой книге содержится много информации, которая поможет начать изучение UNIX. Глава 4 представляет собой учебник по основам UNIX, а во второй части книги содержатся сведения по администрированию Linux. Возможно, лучше прочесть эти главы до того, как вообще начинать установку Linux. Содержащиеся в них сведения могут оказаться очень ценными, если придется столкнуться с какими-либо проблемами.

Не следует рассчитывать на превращение новичка в системного администратора UNIX за одну ночь. Ни одна операционная система не заработает без проблем, и любая из них требует сопровождения, поэтому нужно быть готовым к сложностям. Рассматривайте это как возможность лучше изучить Linux и UNIX и не теряйте мужества, если не все будет так гладко, как хотелось бы!

Советы опытным пользователям UNIX

Даже тем, у кого за плечами опыт многих лет программирования и системного администрирования UNIX, может понадобиться содействие, чтобы собрать и установить Linux. Знатокам UNIX следует ознакомиться с некоторыми аспектами Linux, прежде чем пускаться в плавание. Во-первых, Linux не является коммерческой UNIX-системой. Она не пытается следовать тем же стандартам, что и другие UNIX-системы, с которыми вы могли сталкиваться, но в известном смысле Linux *изменяет* мир UNIX, заставляя все прочие системы потрудиться за те деньги, которых они стоят. Если быть более точным, стабильность является очень важным фактором, но не *единственным*.

Во-вторых, далеко не последнее значение имеет функциональность. Во многих случаях новый программный код включается в ядро, даже если в нем все еще есть ошибки и не все функции реализованы. При этом предполагается, что важнее выпустить код, который пользователи смогут протестировать и которым они смогут пользоваться, чем задерживать выпуск новой версии, пока она не достигнет «совершенства». Почти все проекты open source проходят стадию альфа-версии, прежде чем они полностью готовы к эксплуатации. В результате сообщество open source в целом имеет возможность работать с кодом, тестировать и развивать его, а те, кто находит альфа-версию «достаточно хорошей» для своих нужд, могут ею пользоваться. Коммерческие производители UNIX редко занимают такую позицию при выпуске программного обеспечения, если вообще когда-либо делают это.

Даже если вы суперспециалист по UNIX и можете дизассемблировать ядро Solaris во сне или перекодировать суперблок AIX с руками, связанными за спиной, вам все равно придется некоторое время привыкать к Linux. Это современная и динамичная система, новые версии ядра которой выходят каждые несколько месяцев,

а новые утилиты появляются постоянно. Сегодня ваша система вполне современна, а завтра вы оказываетесь с ней в каменном веке.

Как же при такой динамичности не отстать в изменчивом мире Linux? По большей части обновление лучше производить пошагово, то есть обновлять те части системы, которые *требуют* обновления, и только тогда, когда вы считаете обновление необходимым. Например, если вы никогда не пользуетесь редактором Emacs, нет смысла устанавливать в системе каждую новую его версию. Более того, даже если вы активно используете Emacs, обычно нет необходимости обновлять его, пока не обнаружится отсутствие какой-то функции, имеющейся в новой версии. Нет причин всегда стремиться к обладанию самой последней версией программы.

Помните, Linux разрабатывается своими пользователями. Из этого следует, что чаще всего Linux поддерживает ту аппаратуру, с которой приходится сталкиваться ее пользователям и разработчикам. Это приводит к тому, что поддерживается большая часть популярной аппаратуры и периферийных устройств для систем, основанных на 80x86 (Linux, возможно, поддерживает больше аппаратных устройств, чем любая коммерческая реализация UNIX). Однако отсутствует поддержка некоторых темных и таинственных устройств, как и тех, изготовители которых поставляют фирменные драйверы, не раскрывая спецификаций.[1] С течением времени спектр поддерживаемой аппаратуры постоянно расширяется, поэтому, если вы не найдете своего любимого устройства в списке, не исключено, что его поддержка появится в скором будущем.

Другая трудность поддержки аппаратных устройств в Linux обусловлена тем, что многие компании решили сделать интерфейсы производимой ими аппаратуры своей собственностью.[2] В итоге добровольные разработчики Linux просто не могут написать драйверы для таких устройств (если бы они это сделали, владельцами драйверов стали бы компании, которые владеют интерфейсами, что нарушило бы условия GPL). Компании, сделавшие интерфейсы своей собственностью, пишут собственные драйверы для операционных систем, таких как Microsoft Windows. Конечному пользователю (то есть вам) ничего не требуется знать об интерфейсе. К сожалению, это не дает возможности разработчикам Linux писать драйверы для этих устройств.

Едва ли можно что-то изменить в этой ситуации. Иногда программисты пытаются писать хакерские драйверы исходя из догадок о том, каков может быть интерфейс. В других случаях разработчики пытаются работать с соответствующей компанией, пытаясь получить информацию об интерфейсе устройства, с разной степенью успеха.[3]

---

[1] Чаще всего нежелание публиковать детальные спецификации на новые модели оборудования присуще производителям видеоадаптеров: к настоящему времени многие популярные и уже широко используемые модели видеоадаптеров так и не имеют доступной опубликованной спецификации. – *Примеч. науч. ред.*

[2] То есть запатентовали саму спецификацию протокола. – *Примеч. науч. ред.*

[3] Именно на этом пути в последние несколько лет достигнуты большие успехи: видя широкое распространение Linux и его положительную динамику, даже крупнейшие (брендовые) изготовители стали предоставлять разработчикам драйверов на условиях конфиденциальности детальные спецификации на самые новые модели оборудования для написания драйверов. – *Примеч. науч. ред.*

Источники информации по Linux

Как вы, вероятно, догадываетесь, помимо этой книги существует масса источников информации о Linux.

Электронные документы

При наличии выхода в Интернет через Web или с анонимных сайтов FTP, разбросанных по всему свету, можно получить доступ к большому объему документации по Linux. Даже при отсутствии прямого доступа к Интернету все же существует возможность получить эти документы, поскольку многие дистрибутивы Linux, распространяющиеся на компакт-дисках, содержат все необходимые документы, а кроме того, их часто можно купить в розницу.

В Интернете существует масса веб-сайтов и FTP-архивов с программным обеспечением для Linux и сопутствующей документацией. В приложении перечислены некоторые источники информации о Linux, имеющиеся в Интернете.

Примерами существующих электронных документов о Linux могут служить: Linux FAQ – сборник часто встречающихся вопросов о Linux; документы Linux HOWTO, каждый из которых описывает отдельный аспект системы, в том числе «Installation HOWTO» (установка), «Printing HOWTO» (печать) и «Ethernet HOWTO» (локальные сети). Linux META-FAQ – список иных информационных ресурсов Linux в Интернете.

Кроме этого в Интернете можно найти целые веб-сайты, содержащие разного рода HOWTO, блоги, базы знаний, и форумы, которые предоставляют значительный объем информации, способный оказать помощь желающим пользоваться операционной системой Linux. Производители дистрибутивов поддерживают разнообразные списки рассылки и форумы, где обсуждаются, например, вопросы, имеющие отношение к использованию Linux на ноутбуках или к настройке серверов сети. Такие веб-сайты и каталоги списков рассылки во многом приняли эстафету от телеконференций Usenet, связанных с тематикой Linux. Подробнее о Usenet мы поговорим в разделе «Телеконференции Usenet» далее в этой главе.

Главная страница сервера Linux Documentation находится по адресу *http://www.tldp.org*. Здесь можно найти большое количество HOWTO и других документов, а также ссылки на другие сайты, интересные для пользователей Linux, в том числе на руководства из Проекта документирования Linux – Linux Documentation Project (подробнее об этом в следующем раздел).

Книги и другие печатные издания

Существует великое множество печатных изданий, специально посвященных Linux. Помимо этого в Интернете можно найти большое число свободно распространяемых книг, опубликованных проектом документирования Linux (Linux Documentation Project, LDP). Этот проект был создан в Интернете специально, чтобы создавать и распространять добротный набор руководств для Linux. Эти руководства являются аналогами комплектов документации, распространяемых в составе коммерческих версий UNIX. Они охватывают полный спектр тем от установки Linux до ее использования, программирования, работы с сетями, разработки ядра и т. п.

Руководства LDP можно получить через Web, а также заказать по почте из различных источников. Издательство O'Reilly опубликовало руководство «Linux Network Administrator's Guide» из проекта LDP.[1]

Помимо растущего числа книг по Linux существует множество книг по UNIX, которые, конечно, применимы к Linux: в том, что касается использования и программирования системы, Linux в большинстве случаев не сильно отличается от других реализаций UNIX. Вооружившись несколькими хорошими книгами по UNIX и книгой, которую сейчас держите в руках, вы сможете справиться почти с любыми проблемами.

Существующие ежемесячные журналы, которые дают прекрасную возможность оставаться в курсе последних событий, происходящих в сообществе Linux, особенно это относится к «Linux Journal» и «Linux Magazine». Существуют печатные издания, выходящие и на других языках, помимо английского. (Европейские, южноамериканские и азиатские издания за последние несколько лет перестали быть редкостью.)

Телеконференции Usenet

Usenet является всемирным форумом электронных новостей и дискуссий с большим количеством так называемых телеконференций – дискуссионных клубов, посвященных какой-либо избранной теме. Значительная часть разработки Linux прошла по волнам Интернета и Usenet, и неудивительно, что существует такое количество телеконференций, посвященных Linux.

Телеконференций, посвященных Linux, слишком много, чтобы можно было здесь перечислить их. Те, которые непосредственно относятся к Linux, входят в иерархию *comp.os.linux*, но есть и другие по родственным темам, например *comp.windows.x*.

Списки рассылки в Интернете

При наличии доступа к электронной почте можно принять участие в ряде списков рассылки. Спектр обсуждаемых в них тем простирается от программирования ядра до основных вопросов, возникающих у пользователей. Многие популярные почтовые списки имеют веб-сайты, хранящие архивы, в которых можно вести поиск, что позволяет легко найти ответы на часто задаваемые вопросы. Часть этих ресурсов перечислена в приложении.

Получение помощи

В первую очередь следует упомянуть, что Linux имеет огромное сообщество сторонников, которые сами нуждаются в помощи и готовы совершенно бесплатно предложить свою помощь. Яркий пример такого сообщества – Ubuntu (*http://www.ubuntulinux.org*). Поддерживаемое коммерческой компанией Canonical Ltd., оно готово предложить недорогую профессиональную поддержку. Дистрибутив

[1] Олаф Кирк «Linux для профессионалов. Руководство администратора сети». – Пер. с англ. – СПб.: Питер, 2000.

Ubuntu имеет большое и сплоченное сообщество, готовое предоставить помощь. Ubuntu, выросший из Debian, нанимает множество оплачиваемых разработчиков, которые одновременно помогают поддерживать проект Debian.

Производители таких дистрибутивов, как Red Hat, Novell SUSE и Mandriva, обеспечивают весьма профессиональную поддержку своих собственных дистрибутивов на коммерческой основе.

Следуя концепции, изначально высказанной Бернардом Голденом (Bernard Golden), которую он назвал «Модель зрелости open source», компании, ведущие разработку Linux, демонстрируют свою конкурентоспособность, используя парадигму open source. Они в состоянии обеспечить:

- Адекватную поддержку и обслуживание.
- Ведение новых разработок.
- Постановку целей дальнейшего развития своего продукта и их достижение.
- Широту функциональных возможностей и простоту в эксплуатации, особенно для промышленных ИТ-специалистов.
- Устойчивость бизнес-модели и возможность финансировать новые разработки с охватом новых для себя областей.
- Стройную и расширяемую экосистему партнерства, главная цель которой – достижение успеха клиентами.

Кроме того, эти компании участвуют в проектах сообщества, что предохраняет их от застоя.

Зрелые Linux-компании предоставляют дополнительные коммерческие предложения, включая обучение, продажи, техническую поддержку в формате 24 часа в сутки, 7 дней в неделю, 365 дней в году, компенсации и качественную документацию.

В дополнение к уже упомянутым компаниям вы можете найти немало их деловых партнеров, которые обладают серьезным опытом в обеспечении коммерческой поддержки Linux. Их веб-сайты содержат указания на своих партнеров, благодаря которым их легко можно отыскать и обратиться за помощью тем или иным способом.

Поскольку вы будете становиться все более подготовленными к работе с Linux, вероятно, вы обнаружите массу аспектов, способных приятно удивить. Многие не только используют Linux, но и считают сообщество своим домом. Удачи вам в будущем.

2

Подготовка к установке и установка

Эта глава представляет ваш первый шаг в установке Linux. Мы расскажем, как получить программное обеспечение для Linux в виде одного из укомплектованных дистрибутивов, как подготовить систему к установке, и как создать дисковые разделы для установки Linux на компьютере, где уже была установлена Windows или какая-нибудь другая операционная система.

Как уже говорилось, не существует единого «официального» дистрибутива Linux. На самом деле существует много дистрибутивов, каждый из которых служит определенной цели и набору задач. Эти дистрибутивы можно получить с анонимного FTP-сервера в Интернете, по почте в виде CD-ROM или DVD либо купить в магазине.

Дистрибутивы Linux

Поскольку Linux является свободно распространяемым программным обеспечением, нет единой организации или лица, ответственного за ее выпуск и распространение. Поэтому любой желающий может собрать вместе и распространять программное обеспечение Linux, если при этом не нарушаются ограничения, накладываемые лицензией GPL (или других лицензий, которые могут использоваться). Результатом такого положения дел является наличие большого количества дистрибутивов Linux, распространяемых через анонимные FTP-серверы или по почте.

Теперь перед вами встает задача выбора дистрибутива, который наилучшим образом подходит для ваших нужд. Между дистрибутивами существуют определенные различия. Многие из них содержат практически все программы, которые нужны, чтобы установить полную систему, и даже больше. Другие дистрибутивы – «маленькие» и рассчитаны на пользователей со скромным объемом дискового пространства.

Нужно также учесть, что дистрибутивы могут быть нацелены на разные группы пользователей. Одни предназначены, скорее, для предприятий, другие – для использования на домашних компьютерах. Одни более пригодны для установки на серверы, другие – для создания рабочих станций.

Как сделать правильный выбор среди такого разнообразия дистрибутивов? Если у вас есть доступ к телеконференциям Usenet или другой системе конференций, можно спросить личное мнение тех, кто уже установил Linux. Еще лучше обратиться за помощью и советом к своим знакомым, уже установившим Linux. На практике большинство распространенных дистрибутивов Linux содержит примерно одинаковый набор программ, поэтому можно выбирать дистрибутив достаточно произвольно.

Особый интерес могут представлять дистрибутивы, не требующие установки и запускающиеся прямо с компакт-диска (так называемые *live CD*), такие как Knoppix (*http://www.knoppix.org*). Такие дистрибутивы после загрузки хранят всю информацию в оперативной памяти, но при этом имеют возможность обращаться к жестким дискам и другой аппаратуре компьютера. Кроме того, они позволяют опробовать дистрибутив без необходимости устанавливать его и служат прекрасным инструментом восстановления системы после фатального сбоя. Более подробно о восстановлении системы мы поговорим ниже, в этой же книге.

Получение носителя с Linux по почте или иным образом

При отсутствии высокоскоростного доступа к Интернету можно получить дистрибутив Linux, заказав CD-ROM или DVD по почте. Многие компании, распространяющие дистрибутивы, принимают кредитные карты и заказы из-за рубежа, поэтому существует возможность приобрести Linux вне зависимости от вашего места жительства.

Linux является свободно распространяемым программным обеспечением, но согласно GPL дистрибьюторы могут взимать за нее плату. Поэтому заказ Linux по почте может обойтись вам от 5 до 150 долларов США в зависимости от выбранного дистрибутива. Однако, если у вас есть знакомые, которые приобрели или загрузили Linux, можно позаимствовать или скопировать их программное обеспечение для собственного использования. Производители дистрибутивов не имеют права ограничивать лицензию или дальнейшее распространение программного обеспечения. Например, если вы намерены установить Linux на целую группу машин, достаточно будет приобрести одну копию дистрибутива, который можно использовать для установки Linux на все машины. Есть, впрочем, одно исключение из этого правила: чтобы повысить цену на свои дистрибутивы, некоторые поставщики включают в него коммерческие пакеты, устанавливать которые на нескольких машинах может быть запрещено. В таких случаях пакет должен содержать явные указания на это.

Еще одно преимущество, которое дает покупка дистрибутива, состоит в том, что после приобретения вы получаете техническую поддержку на установку, то есть можете связаться с поставщиком по телефону или по электронной почте и получить квалифицированную помощь в разрешении проблем, которые могут возникнуть в процессе установки.

Многие группы пользователей Linux предлагают свои дистрибутивы. Поинтересуйтесь, нет ли такой группы поблизости от вас. Для отдельных платформ, таких как Alpha, получение дистрибутива Linux от группы пользователей будет наилучшим выбором.

Получение Linux из Интернета

Если у вас есть доступ в Интернет, проще всего будет загрузить Linux с анонимного FTP-сервера. Один из основных серверов FTP – *ftp://ftp.ibiblio.org*, здесь можно найти самые разные дистрибутивы в каталоге */pub/Linux/distributions*. Во многих странах существуют локальные зеркала этого сервера, откуда вы сможете загрузить тот же самый дистрибутив значительно быстрее.

При загрузке Linux обязательно используйте для передачи файлов двоичный режим (в большинстве FTP-клиентов этот режим включается командой *binary*).

Небольшие проблемы могут возникнуть при загрузке файлов одной системы (например, Linux) с помощью другой системы (например, Windows), поскольку эти системы не всегда одинаково работают с файлами других систем. Однако советы, помещенные в этой главе, должны дать вам возможность благополучно добраться до конца процесса установки.

Некоторые дистрибутивы распространяются через анонимные FTP-серверы в виде набора образов дисков. Это значит, что дистрибутив состоит из ряда файлов, каждый из которых содержит двоичный образ гибкого диска. Чтобы скопировать двоичный образ на дискету, можно использовать программу *RAWRITE.EXE* под Windows. Эта программа поблочно копирует содержимое файла на дискету, игнорируя его формат. *RAWRITE.EXE* можно взять на многих FTP-серверах Linux, в частности на *ftp://ftp.ibiblio.org* в каталоге */pub/Linux/system/Install/rawwrite*.

Сразу предупреждаем, что это трудоемкий способ установки Linux: дистрибутив вполне может насчитывать более 50 дисков. При таком способе нужно загрузить набор образов гибких дисков и записать их с помощью *RAWRITE.EXE* на дискеты. Вследствие этого лишь немногие дистрибутивы распространяются исключительно в виде образов дискет. Однако комбинация из нескольких дискет, предназначенных для выполнения процедуры начальной загрузки, плюс один или более компакт-дисков, встречается не так уж и редко.

После загрузки компьютера с так называемого *загрузочного диска* (*boot floppy*) можно приступать к установке. Установка обычно производится непосредственно с гибких дисков, хотя некоторые дистрибутивы позволяют осуществлять ее с раздела Windows на жестком диске, а другие дают возможность производить установку по сети TCP/IP. В каждом дистрибутиве должна быть документация, описывающая эти методы установки, если они возможны.

Если у вас есть UNIX-станция с дисководом для гибких дисков, можно прямо копировать файл образа на дискету с помощью команды *dd*. Например, на рабочей станции Sun команда `dd of=/dev/rfd0 if=foo bs=18k` произведет «сырую запись» содержимого файла *foo* на гибкий диск. Проконсультируйтесь у своих местных знатоков UNIX относительно гибких дисков в вашей системе и использования *dd*.

Каждый дистрибутив Linux, полученный с анонимного сервера FTP, должен содержать файл *README*, в котором описана процедура загрузки и подготовки гибких дисков для установки. Обязательно прочтите всю документацию по используемой вами версии дистрибутива.

В последнее время некоторые крупные дистрибутивы Linux распространяются в виде ISO-образов, которые можно прожечь на CD-ROM или DVD. Ввиду огром-

ного объема данных загрузка таких образов возможна только при наличии большого жесткого диска и интернет-соединения с большой пропускной способностью (в принципе, место на жестком диске потребуется всего лишь для одного ISO-образа, поскольку после того, как он будет записан на компакт-диск, можно удалить его и затем загрузить следующий образ).

Подготовка к установке Linux

После получения дистрибутива нужно подготовить машину к установке Linux. Этот процесс требует некоторого планирования, особенно если у вас уже установлена другая ОС. В следующих разделах мы расскажем, как планировать установку Linux.

Общие сведения об установке

Хотя все дистрибутивы Linux различны, общий метод установки следующий:

1. *Заново разбейте жесткий диск (диски) на разделы.* Если у вас уже установлены другие ОС, вам понадобится заново разбить диски, чтобы освободить место для Linux. Эта процедура описана в разделе «Принципы создания разделов» далее в этой главе. В некоторых дистрибутивах (таких как SUSE) это действие интегрировано в процедуру установки. Проверьте в документации, так ли это в вашем случае. В любом случае не помешает разбить жесткий диск на разделы заранее.
2. *Загрузитесь с установочного диска.* У каждого дистрибутива есть установочный носитель, которым обычно являются загрузочная дискета или загрузочный компакт-диск. Загрузка с него вызовет некоторую программу инсталляции, которая пошагово проведет вас через процесс установки либо позволит осуществить установку вручную.
3. *Создайте разделы Linux.* После переразбиения диска с целью выделения места под Linux нужно создать в этом свободном пространстве разделы Linux. Это выполняется с помощью программы Linux *fdisk*, о которой рассказано в разделе «Редактирование /etc/fstab», либо с помощью другой специфической для данного дистрибутива программы, такой как *Disk Druid*, поступающей с Red Hat Linux.
4. *Создайте файловые системы и раздел подкачки (свопинга).* На этом этапе на вновь созданных разделах создаются одна или несколько файловых систем для хранения файлов. Кроме того, если вы собираетесь использовать свопинг (а это необходимо, если только на вашей машине не установлена оперативная память гигантского размера), нужно выделить пространство для свопинга на одном из разделов Linux. Это описано в разделах «Создание пространства для свопинга» и «Редактирование /etc/fstab».
5. *Установите программное обеспечение на новые файловые системы.* Наконец, вы устанавливаете программное обеспечение на вновь созданные файловые системы. Об этом рассказывается в разделе «Установка программного обеспечения». Далее в разделе «Устранение неполадок» мы расскажем, как действовать, если что-то не получается.

Те, кто хочет переключаться между различными ОС, иногда находятся в сомнении, какую систему устанавливать сначала – Linux или другую? Мы свидетельствуем, что при установке Windows 95/98/ME после Linux могут возникнуть неприятности. Windows 95/98/ME имеет привычку при установке затирать данные в загрузочном секторе, поэтому безопаснее сначала установить Windows, а затем Linux, используя сведения, приведенные в этой главе. Похоже, что Windows NT/2000/XP ведет себя более терпимо в отношении уже существующих данных в загрузочном секторе, но безопаснее будет сначала установить Windows.

Во многих дистрибутивах Linux есть программа установки, которая проводит вас через процесс установки и автоматизирует один или несколько из описанных этапов. Читая эту и следующую главу, помните, что в зависимости от дистрибутива некоторое число описанных выше шагов может быть автоматизировано.

При подготовке к установке Linux очень полезно вести записи в течение всего процесса. Записывайте все, что вы делаете, все, что вводите с клавиатуры, и все, что может показаться вам ненормальным. Идея проста: если (или когда) случится неприятность, вам потребуется отследить свои шаги, чтобы понять, что произошло. Установить Linux несложно, но нужно помнить многие детали. Вам потребуется регистрировать все эти детали, чтобы при необходимости попробовать другие методы. Иметь записи о процессе установки полезно и в том случае, если потребуется обратиться к кому-нибудь за помощью, например послать сообщение в одну из телеконференций по Linux. А возможно, вы когда-нибудь захотите показать свой дневник внукам.[1]

Принципы создания разделов

Как правило, жесткие диски разбиваются на *разделы* (*partitions*), и каждой операционной системе отводится один или несколько разделов. Например, на одном диске можно иметь несколько разделов: один, скажем, для Windows, другой для FreeBSD и еще два для Linux.

Если у вас на компьютере уже установлено какое-то программное обеспечение, вам может потребоваться изменить размеры разделов, чтобы освободить место для Linux. Затем вы создадите на освободившемся месте один или несколько разделов Linux для хранения программного обеспечения Linux и свопинга. Мы называем этот процесс *переразбиением* (*repartitioning*).

Во многих системах Windows использует единственный раздел, занимающий весь диск. В Windows этот раздел называется `C:`. Если разделов несколько, Windows называет их `D:`, `E:` и т. д. В некотором смысле каждый раздел ведет себя как отдельный жесткий диск.

В первом секторе диска находятся *главная загрузочная запись* (*master boot record*) и *таблица разделов* (*partition table*). Главная загрузочная запись, как и го-

[1] Мэтт стыдливо признается, что записывал в дневник все свои неприятности с Linux в течение первых нескольких месяцев работы с системой. Сейчас дневник пылится на полке.

ворит ее название, используется для загрузки системы. Таблица разделов содержит данные о расположении и размерах дисковых разделов.

Есть три вида разделов: *первичный* (*primary*), *расширенный* (*extended*) и *логический* (*logical*). Чаще всего используются первичные разделы. Однако, поскольку размер таблицы разделов ограничен, на каждом жестком диске можно иметь только четыре первичных раздела. Это следствие плохой продуманности MS-DOS и Windows, в других ОС, даже относящихся к той же эпохе, таких ограничений нет.

Обойти ограничение на четыре раздела можно с помощью расширенного раздела. Расширенный раздел сам по себе не содержит никаких данных, а служит контейнером для логических разделов. Поэтому можно создать расширенный раздел, охватывающий весь диск, а в нем создать много логических разделов. Однако на каждом диске можно создать только один расширенный раздел.

Требования Linux к дисковым разделам

Прежде чем выяснять, как произвести переразбиение дисков на разделы, нужно получить представление о том, сколько места требуется для Linux. О том, как создавать эти разделы, мы расскажем позже в разделе «Редактирование /etc/fstab».

В UNIX-системах файлы хранятся в файловой системе, которая, в сущности, является частью жесткого диска (или другого носителя, например компакт-диска, DVD или гибкого диска), отформатированной для хранения файлов. Каждая файловая система связана с конкретной частью дерева каталогов. Например, во многих системах существует файловая система для всех файлов в каталоге */usr*, другая файловая система для каталога */tmp* и т. д. *Корневая* (*root*) файловая система является главной и соответствует самому верхнему каталогу /.

В Linux каждая файловая система существует в отдельном разделе жесткого диска. Например, если есть одна файловая система для каталога / и другая для */usr*, нужны два дисковых раздела для хранения этих двух файловых систем.[1]

Перед установкой Linux нужно подготовить файловые системы для хранения программного обеспечения. У вас должна быть как минимум одна файловая система (корневая), а значит, один раздел диска, отведенный под Linux. Многие пользователи Linux решают хранить все свои файлы в корневой файловой системе, что во многих случаях проще, чем разбираться с несколькими файловыми системами и разделами.

Однако при желании можно создать для Linux несколько файловых систем.[2] Например, можно использовать разные файловые системы для */usr* и */home*. Читатели, имеющие опыт администрирования UNIX-систем, знают, как можно творчески подойти к использованию нескольких файловых систем. В разделе «Соз-

---

[1] Обратите внимание: это относится к файловым системам, но не к каталогам. Само собой разумеется, в пределах одной и той же файловой системы внутри корневого каталога может находиться любое число подкаталогов.

[2] Но их можно добавить и гораздо позже к единой корневой файловой системе */;* это практически не затронет работоспособность системы. – *Примеч. науч. ред.*

дание файловых систем» главы 10 мы обсудим использование нескольких разделов и файловых систем.

Для чего может понадобиться несколько файловых систем? Чаще всего они организуются по соображениям надежности: если по какой-либо причине одна из файловых систем оказывается поврежденной, остальные (обычно) остаются в целости. Напротив, если хранить все файлы в корневой файловой системе, то при ее повреждении теряются сразу все файлы. Однако это случается довольно редко. Если регулярно проводить резервное копирование, то можно чувствовать себя в безопасности.

С другой стороны, использование нескольких файловых систем дает то преимущество, что можно легко обновить систему, не подвергая опасности свои бесценные данные. Можно завести раздел для личных каталогов пользователей и при обновлении системы, не трогая этого раздела, очистить остальные и установить Linux «с нуля». Конечно, в современных дистрибутивах есть развитые процедуры обновления, но иногда возникает желание начать все сначала.

Несколько файловых систем используют также для того, чтобы распределить дисковую память между несколькими жесткими дисками. Если у вас, скажем, свободны 300 Мбайт на одном жестком диске и 2 Гбайт – на другом, то можно создать 300-мегабайтную корневую файловую систему на первом диске и 2-гигабайтную файловую систему */usr* – на втором. Существует также возможность охватить одной файловой системой несколько дисков, для чего используется так называемый *диспетчер логических томов* (*Logical Disk Manager*, *LVM*), но его сложно настроить, если входящая в дистрибутив программа установки не делает этого автоматически.

Итак, для Linux требуется, по крайней мере, один раздел – для корневой файловой системы. Если вы хотите создать несколько файловых систем, для каждой из дополнительных файловых систем нужен свой раздел на диске. Некоторые дистрибутивы Linux автоматически создают разделы и файловые системы, так что вообще не приходится беспокоиться об этих проблемах.

При планировании разделов следует также подумать о свопинге. *Пространство свопинга* (*swap space*) – это часть диска, используемая системой для временного хранения фрагментов программ, загруженных пользователем, но не используемых в данный момент. Linux не требует обязательного выделения пространства для свопинга, но если объем установленной в машине физической оперативной памяти меньше 256 Мбайт, настоятельно рекомендуется организовать свопинг.

Есть две возможности. Первая – использовать *файл свопинга*, располагаемый в одной из файловых систем Linux. Файл свопинга для использования в качестве виртуальной памяти создается после установки программного обеспечения. Вторая возможность – создать *раздел для свопинга* (*подкачки*) в качестве отдельного раздела, используемого только с этой целью. В большинстве случаев для свопинга используется раздел, а не файл.[1]

---

[1] Раздел свопинга – быстрее, поскольку ОС пишет по не размеченному файловой системой пространству, но файл свопинга может оказаться более гибким в использовании. – *Примеч. науч. ред.*

Отдельный файл или раздел для свопинга может занимать до 2 Гбайт.[1] При желании использовать для свопинга более 2 Гбайт (что едва ли может потребоваться) можно создать несколько файлов или разделов свопинга общим количеством до 32.

Предположим, что вы хотите запустить утилиту *fdisk*, чтобы переразбить первый SCSI-диск, тогда для этого нужно запустить следующую команду:

```
# fdisk /dev/sda
```

Если не указать диск, то значением по умолчанию будет */dev/hda* (первый диск IDE).

Если вы создаете разделы Linux на нескольких дисках, запустите *fdisk* по одному разу для каждого диска:

```
# fdisk /dev/hda

Command (m for help):
```

Теперь *fdisk* ждет команду; можно ввести `m`, чтобы получить список возможных команд:

```
Command (m for help): m
Command action
   a   toggle a bootable flag
   b   edit bsd disklabel
   c   toggle the dos compatibility flag
   d   delete a partition
   l   list known partition types
   m   print this menu
   n   add a new partition
   o   create a new empty DOS partition table
   p   print the partition table
   q   quit without saving changes
   s   create a new empty Sun disklabel
   t   change a partition's system id
   u   change display/entry units
   v   verify the partition table
   w   write table to disk and exit
   x   extra functionality (experts only)

Command (m for help):
```

Команда `n` используется для создания нового раздела. Большая часть остальных команд вам не понадобится. Чтобы завершить работу *fdisk* без сохранения изменений, воспользуйтесь командой `q`. Чтобы выйти из *fdisk* и записать измененную таблицу разделов на диск, воспользуйтесь командой `w`. Стоит повторить: если вы закончите работу командой `q` без записи, можете делать в *fdisk* что хотите, не рискуя потерять данные. Только при вводе `w` может произойти катастрофа, если вы в чем-то ошибетесь.

---

[1] Эта величина относится только к процессорам Intel. В других архитектурах она может быть как больше, так и меньше.

Первым делом нужно вывести текущую таблицу разделов и записать результат, который может в дальнейшем понадобиться. Это можно сделать с помощью команды p. Неплохо записывать в тетрадь каждое изменение, проведенное в таблице разделов. Если по какой-либо причине таблица разделов будет повреждена, данные на диске станут недоступными, несмотря на то, что они останутся на месте. Но с помощью своих записей во многих случаях можно восстановить таблицу разделов и вернуть данные, если, снова запустив *fdisk*, удалить и воссоздать разделы с теми параметрами, которые вы записали ранее. Не забудьте по окончании работы сохранить восстановленную таблицу разделов.

Вот пример вывода таблицы разделов с указанием блоков, секторов и цилиндров, описывающих структуру диска:

```
Command (m for help): p

Disk /dev/hda: 16 heads, 38 sectors, 683 cylinders
Units = cylinders of 608 * 512 bytes
   Device Boot  Begin  Start    End  Blocks   Id  System
/dev/hda1   *       1      1    203  61693     6  DOS 16-bit >=32M

Command (m for help):
```

В этом примере есть один раздел Windows на */dev/hda1*, в котором 61 693 блока (около 60 Мбайт).[1] Раздел начинается с цилиндра 1 и заканчивается цилиндром 203. Всего на этом диске 683 цилиндра, поэтому остается 480 цилиндров, на которых можно создать разделы Linux.

Чтобы создать новый раздел, воспользуйтесь командой n. В этом примере мы создадим для Linux два новых первичных раздела (*/dev/hda2* и */dev/hda3*):

```
Command (m for help): n
Command action
   e  extended
   p  primary partition (1-4)
   p
```

Здесь *fdisk* спрашивает, какого типа раздел нужно создать – расширенный или первичный. В нашем примере создаются только первичные разделы, поэтому мы выбираем p:

```
 Partition number (1-4):
```

Затем *fdisk* спрашивает номер создаваемого раздела; поскольку раздел 1 уже есть, первый раздел Linux будет иметь номер 2:

```
Partition number (1-4): 2
First cylinder (204-683):
```

Теперь мы укажем номер начального цилиндра раздела. Поскольку не использованы цилиндры с 204 по 683, мы используем первый свободный цилиндр (с номером 204). Нет смысла оставлять пустое пространство между разделами:

```
First cylinder (204-683): 204
Last cylinder or +size or +sizeM or +sizeK (204-683):
```

[1] Размер одного блока в Linux составляет 1024 байт.

fdisk запрашивает размер создаваемого раздела. Можно задать номер последнего цилиндра либо размер в байтах, килобайтах или мегабайтах. Поскольку мы хотим создать раздел размером 80 Мбайт, укажем `+80M`. При таком задании размера раздела *fdisk* округляет фактический размер раздела до ближайшего числа цилиндров:

```
Last cylinder or +size or +sizeM or +sizeK (204-683): +80M

Warning: Linux cannot currently use 33090 sectors of this partition
```

(Предупреждение: Linux в настоящее время не может использовать 33 090 секторов этого раздела.)

Если вы увидите такое сообщение, можете его проигнорировать. *fdisk* выводит его, потому что это старая программа из тех времен, когда разделам Linux не позволялось быть больше 64 Мбайт.

Теперь можно создать второй раздел Linux. В целях наглядности сделаем его размером в 10 Мбайт:

```
Command (m for help): n
Command action
  e extended
  p primary partition (1-4)
p
Partition number (1-4): 3
First cylinder (474-683): 474
Last cylinder or +size or +sizeM or +sizeK (474'683): +10M
```

Наконец, выведем таблицу разделов. Опять запишем все эти данные, особенно число блоков в новых разделах. Размеры разделов понадобятся при создании файловых систем. Убедитесь также, что разделы не перекрывают друг друга:

```
Command (m for help): p

Disk /dev/hda: 16 heads, 38 sectors, 683 cylinders
Units = cylinders of 608 * 512 bytes
   Device  Boot  Begin  Start   End  Blocks   Id  System
/dev/hda1    *       1      1   203   61693    6  DOS 16-bit >=32M
/dev/hda2          204    204   473   82080   83  Linux native
/dev/hda3          474    474   507   10336   83  Linux native
```

Как видно, */dev/hda2* является теперь разделом размером 82 080 блоков (что соответствует примерно 80 Мбайт), а */dev/hda3* содержит 10 336 блоков (около 10 Мбайт).

Обратите внимание, что в большинстве дистрибутивов необходимо выполнить в *fdisk* команду `t`, чтобы изменить тип раздела для свопинга на Linux swap, который имеет номер 82. С помощью команды `l` можно вывести список кодов известных типов разделов, а затем командой `t` установить тип раздела для свопинга в тот, который соответствует Linux swap.

Благодаря этому программа установки сможет автоматически найти разделы для свопинга по типу. Если окажется, что программа установки не может найти раздел для свопинга, нужно снова запустить *fdisk* и установить тип нужного раздела командой `t`.

В нашем примере оставшиеся на диске цилиндры с номерами от 508 до 683 не используются. Можно оставить на диске свободное пространство, чтобы позднее создать дополнительные разделы.

Наконец, с помощью команды `w` запишем изменения на диск и выйдем из *fdisk*:

```
Command (m for help): w
#
```

Запомните, что все изменения, которые вы производите во время работы с *fdisk*, вступают в силу только после выполнения команды `w`, поэтому можно попробовать другие конфигурации и только в конце сохранить их. Можно также в любой момент с помощью команды `q` прекратить работу с *fdisk* без сохранения изменений. Не забывайте, что нельзя изменять разделы других операционных систем с помощью программы *fdisk* для Linux.

Загрузка Linux с раздела, в котором есть цилиндры с номерами, бо**Ь**шими 1023, может оказаться невозможной. Поэтому постарайтесь создать корневой раздел Linux в диапазоне цилиндров ниже 1024, что почти всегда можно сделать (например, создав маленький корневой раздел на цилиндрах с номерами до 1024). Если по какой-либо причине вы не можете или не хотите этого сделать, можно просто загружать Linux с гибкого диска, использовать для этих же целей специальную возможность восстановления системы, существующую на установочных дисках, или вообще пользоваться дистрибутивом Linux на live CD.

В некоторых дистрибутивах Linux требуется перезагрузить систему после работы *fdisk*, чтобы изменения в таблице разделов вступили в силу до установки программного обеспечения. Более новые версии *fdisk* автоматически обновляют в ядре данные о разделах, и перезагрузка не нужна. Для надежности лучше после работы с *fdisk* перезагрузиться с инсталляционного носителя и только тогда продолжать работу.

Создание пространства для свопинга

Если вы собираетесь использовать для виртуальной памяти раздел свопинга, теперь можно его подготовить.[1] В разделе «Управление пространством свопинга» главы 10 мы расскажем, как подготовить своп-файл, если вы не хотите использовать отдельный раздел.

Во многих дистрибутивах перед установкой программного обеспечения нужно создать и активировать пространство для свопинга. Если у вас мало оперативной памяти, возможно, что для установки придется выделить какой-то объем пространства под свопинг.

Подготовка пространства для свопинга осуществляется командой *mkswap*, которая имеет следующий формат:

```
mkswap -c partition
```

где *partition* – имя раздела для свопинга. Например, если для свопинга выделен раздел */dev/hda3*, нужно выполнить команду

[1] Опять-таки некоторые дистрибутивы Linux создают пространство для свопинга автоматически или через пункт меню при установке.

```
# mkswap -c /dev/hda3
```

В старых версиях *mkswap* требовалось указать размер раздела, что было опасно, поскольку один неверно введенный знак мог разрушить диск логически.

Параметр *-c* требует, чтобы утилита *mkswap* при создании пространства для свопинга проверила раздел на наличие дефектных блоков, то есть участков магнитного носителя, которые не могут правильно хранить данные. На современных дисках это бывает редко, но, если такие блоки есть и вы об этом не знаете, могут возникнуть бесчисленные неприятности. Всегда используйте параметр *-c*, чтобы проверить диск на наличие плохих блоков. Они будут автоматически исключены из использования.

Если для свопинга отведено несколько разделов, команду *mkswap* нужно выполнить с каждым из них.

После форматирования пространства для свопинга нужно дать системе возможность его использовать. Обычно система автоматически активирует пространство для свопинга при загрузке, однако, поскольку вы еще не установили Linux, это нужно сделать вручную.

Команда, активирующая пространство для свопинга, называется *swapon*. Она имеет следующий формат:

```
swapon partition
```

После выполнения команды *mkswap* активируйте пространство для свопинга на разделе */dev/hda3* следующей командой:

```
# swapon /dev/hda3
```

Создание файловых систем

Для того чтобы записывать файлы в разделы Linux, нужно создать на них файловые системы. Создание файловой системы аналогично форматированию раздела в Windows или другой операционной системе. Мы кратко обсуждали файловые системы в разделе «Требования Linux к дисковым разделам» ранее в этой главе.

В Linux поддерживается несколько типов файловых систем. У каждой файловой системы свой формат и набор характеристик, таких как длина имени файла, максимальный размер файла и т. д. Linux поддерживает также несколько типов файловых систем других разработчиков, например файловую систему Windows.

Чаще всего используются *вторая расширенная файловая система – Second Extended Filesystem*, или *ext2fs*, и *третья расширенная файловая система – Third Extended Filesystem*, или *ext3fs*. *ext2fs* и *ext3fs* представляют собой наиболее эффективные и гибкие файловые системы; они допускают имена файлов длиной до 256 символов и размер файлов до 32 Тбайт. В разделе «Типы файловых систем» главы 10 мы расскажем о различных типах файловых систем, доступных в Linux. Для начала мы предлагаем использовать файловую систему *ext3fs*.

Создать файловую систему *ext3fs* можно командой

```
mke2fs -j -c partition
```

где *partition* — это имя раздела. (Обратите внимание: для создания файловых систем *ext2* и *ext3* используется одна и та же команда *mke2fs*, различие заключается в параметре командной строки *-j*. Если указан этот параметр, утилита *mke2fs* создаст журналирующую файловую систему *ext3*.) Например, чтобы создать файловую систему в разделе */dev/hda2*, нужно выполнить команду

```
# mke2fs -j -c /dev/hda2
```

Чтобы использовать в Linux несколько файловых систем, нужно для каждой из них выполнить соответствующую команду *mke2fs*.

Если в этом месте у вас возникли проблемы, см. раздел «Устранение неполадок» далее в этой главе.

Установка программного обеспечения

Теперь вы можете установить на своем компьютере программное обеспечение. В каждом дистрибутиве для этого существуют свои методы. Во многих дистрибутивах есть специальная программа, которая проведет вас через все этапы установки. В других дистрибутивах нужно *монтировать* (*mount*) ваши файловые системы в определенном каталоге (например, */mnt*) и копировать в них программное обеспечение вручную. Дистрибутивы на CD-ROM предоставляют возможность установить часть программного обеспечения на жесткий диск и оставить большую часть на CD-ROM. Часто это называется *«живой файловой системой»* (live filesystem). Такая файловая система удобна для того, чтобы испытать работу в Linux, прежде чем решиться полностью установить ее на диск.

Некоторые дистрибутивы предлагают несколько способов установки программного обеспечения. Например, может быть предложено установить программное обеспечение прямо с раздела Windows на жестком диске, а не с дискет, или даже провести установку по сети TCP/IP через FTP или NFS. Посмотрите подробности в документации к дистрибутиву.

Дистрибутив Slackware, например, требует следующих действий:

1. Создать разделы с помощью *fdisk*.
2. Создать, по желанию, пространство для свопинга с помощью *mkswap* и *swapon* (если у вас меньше 16 Мбайт RAM).
3. Запустить программу *setup* для установки программного обеспечения. *setup* проводит вас по понятной системе меню.

Конкретный способ установки программного обеспечения Linux сильно зависит от дистрибутива.

Разнообразие устанавливаемого программного обеспечения может показаться ошеломляющим. Современные дистрибутивы Linux вполне могут содержать тысячи пакетов, разбросанных по нескольким CD-ROM. Есть три основных метода отбора программных пакетов:

Отбор по назначению машины

Для новичков это самый простой способ отбора. Не нужно думать о том, требуется ли вам тот или иной пакет, просто решите, должна ли ваша Linux-машина выступать в роли рабочей станции, машины разработчика или сетевого маршрутизатора, и программа установки подберет для вас нужные пакеты.

В любом случае вы можете потом уточнить свой выбор вручную или снова вернуться к программе установки.

Отбор пакетов по сериям

При таком механизме отбора все пакеты группируются по сериям, например «сетевое взаимодействие», «разработка программ» или «графика». Вы можете пробежаться по всем сериям и выбрать в них отдельные пакеты. При этом приходится принимать больше решений, чем при выборе по назначению машины, поскольку для каждого пакета нужно решать, нужен он вам или нет, но можно опустить целую серию, если вы уверены, что предлагаемые ею функции вам не нужны.

Выбор пакетов, отсортированных в алфавитном порядке

Этот способ полезен, только если вы заранее знаете, какие пакеты нужно установить. В противном случае вы не увидите леса из-за деревьев.

Выбор одного из этих методов не исключает использования остальных. В большинстве дистрибутивов предлагаются два или более из вышеупомянутых механизмов отбора.

Все же может оказаться затруднительным выбрать нужные пакеты. В хороших дистрибутивах на экран выводится краткое описание каждого пакета, что облегчает выбор, но если вы не чувствуете уверенности, то наш совет таков: если есть сомнения, оставьте пакет в покое! Позднее вы всегда сможете вернуться и добавить его.

В современных дистрибутивах есть отличная функция, так называемое *отслеживание зависимостей*. Существуют пакеты, работающие только тогда, когда установлены некоторые другие пакеты (например, средству просмотра графики могут понадобиться специальные графические библиотеки для импорта файлов). При отслеживании зависимостей программа установки может сообщать о таких зависимостях, позволяя автоматически выбирать нужный вам пакет вместе с теми, от которых он зависит. Если только вы не уверены абсолютно в правильности своих действий, такое предложение следует всегда принимать, иначе пакет может потом оказаться неработоспособным.

Программы установки могут и другими способами помочь вам сделать выбор и избежать ошибок. Например, программа установки может отказаться начать установку, если вы не выберете пакет, который совершенно необходим для запуска системы хотя бы в минимальном объеме (скажем, базовую структуру каталогов). Или она может проверить наличие взаимоисключающих выборов в случаях, когда можно установить один из пакетов, но не оба вместе.

С некоторыми дистрибутивами поставляется большая книга, в которой перечислены все пакеты с краткими описаниями. Неплохо хотя бы просмотреть эти описания, чтобы представлять, что у вас есть в запасе, и не оказаться в положении, когда в результате вашего выбора на машине окажется 25 текстовых редакторов.

Создание загрузочной дискеты или установка GRUB

Каждый дистрибутив предлагает тот или иной способ начальной загрузки Linux после установки программного обеспечения. Во многих случаях программа установки предлагает создать загрузочную дискету, содержащую ядро Linux,

сконфигурированное для использования вновь созданной вами корневой файловой системы. Для начальной загрузки Linux можно использовать эту дискету. После загрузки управление будет передано жесткому диску. В других дистрибутивах загрузочной является сама установочная дискета. Не стоит волноваться, если в вашем компьютере отсутствует накопитель на гибких магнитных дисках (как во многих современных компьютерах), поскольку всегда существуют альтернативные способы загрузки Linux, как, например, загрузка непосредственно с установочного компакт-диска.

Многие дистрибутивы предлагают установить на жесткий диск GRUB – программу, которая помещается в главную загрузочную запись вашего диска. Она может загружать различные операционные системы, в том числе Windows и Linux, и позволяет выбрать одну из них в момент загрузки.

Чтобы успешно установить загрузчик GRUB, нужно сообщить ему ряд сведений о конфигурации жесткого диска, например, в каких разделах какие операционные системы установлены, как загружать каждую операционную систему и т. д. Многие дистрибутивы при установке GRUB стараются «угадать» правильные параметры вашей конфигурации. Изредка случается, что автоматическая установка GRUB ошибается и превращает в руины главную загрузочную запись (хотя едва ли при этом будут также повреждены фактические данные на жестком диске).

Часто бывает лучше воспользоваться загрузочной дискетой, если нельзя сконфигурировать GRUB самостоятельно. Но если вы абсолютно уверены, то можете прибегнуть к автоматической установке GRUB, когда такая возможность предоставляется вашим дистрибутивом.

В разделе «Использование GRUB» главы 7 мы подробно расскажем о том, как установить и настроить GRUB в вашем конкретном случае.

Помимо GRUB существуют другие начальные загрузчики, например более старый Linux Loader (LILO). Основная концепция этого загрузчика та же самая, но установка и настройка существенно отличаются.

Если все прошло хорошо, примите поздравления! Вы только что установили Linux на свою машину. Выпейте чашечку чаю или чего-нибудь еще, вы это заслужили.

Если вы столкнулись с какими-либо неприятностями, то в разделе «Устранение неполадок» далее в этой главе описаны наиболее часто встречающиеся затруднения при установке Linux и способы их преодоления.

Дополнительные процедуры установки

В некоторых дистрибутивах Linux есть дополнительные процедуры установки, позволяющие сконфигурировать различные пакеты, например сетевые программы TCP/IP, X Window System и т. д. Если во время установки вам предоставляется такая возможность, то далее в этой книге можно прочесть, как настраивать эти программы. В противном случае следует отложить эти процедуры установки до тех пор, пока у вас не возникнет полное понимание того, как настраивать это программное обеспечение.

Решайте сами. Если нет иного выхода, отдайтесь течению и посмотрите, что из этого выйдет. Допустив на этом этапе какие-либо ошибки, вы, скорее всего, сможете исправить их в будущем (постучите по дереву).

Послеустановочные процедуры

После завершения установки программного обеспечения Linux вы должны быть в состоянии перезагрузить машину, зарегистрироваться как *root* и начать изучение системы. (В каждом дистрибутиве для этого есть свой способ, следуйте инструкциям в дистрибутиве.)

Прежде чем начать самостоятельную работу, нужно сделать несколько вещей, чтобы уберечь себя от будущих невзгод. Некоторые из этих задач тривиальны, если у вас подходящие аппаратная часть и дистрибутив Linux. Другие задачи потребуют от вас некоторых исследований, и вы можете отложить их на будущее.

Создание учетной записи пользователя

Чтобы начать использование системы, необходимо создать для себя учетную запись (user account). Если вы предполагаете, что на вашей машине будут работать другие пользователи, нужно будет создать учетные записи и для них. Но для начала нужна хотя бы одна учетная запись.

Для чего это нужно? В каждой системе Linux есть несколько предустановленных учетных записей, например *root* (*суперпользователь*). Однако учетная запись root предназначена исключительно для административных целей. Суперпользователь обладает всевозможными привилегиями, и ему доступны все файлы в системе.

Но работать в системе как пользователь *root* опасно, особенно если у вас нет опыта работы в Linux. Поскольку нет никаких ограничений на действия суперпользователя, очень легко неправильно ввести команду, случайно удалить файлы, повредить файловую систему и т. д. Работать как *root* следует только при выполнении задач системного администрирования, например при изменении файлов настроек, установке нового программного обеспечения и т. д. Подробности можно прочесть в разделе «Сопровождение системы» главы 10.[1]

Для обычной работы нужно создать стандартную учетную запись пользователя. UNIX-системы располагают встроенной системой защиты, которая предотвращает удаление файлов других пользователей и повреждение важных ресурсов, таких как системные файлы конфигурации. Работая как обычный пользователь, вы защитите себя от собственных ошибок. Это особенно относится к пользователям, не имеющим опыта администрирования UNIX-систем.

Во многих дистрибутивах Linux есть средства для создания новых учетных записей. Обычно эти программы имеют название *useradd* или *adduser*. Зарегистрировавшись как *root* и вызвав одну из этих команд, вы получите справку по применению команды, и создание учетной записи должно оказаться вполне очевидным.

---

[1] Попутное замечание: в системах Windows 95/98/ME пользователь всегда обладает правами, равносильными правам *root*, независимо от того, требуются ли такие права в действительности.

В большинстве современных дистрибутивов есть общий инструмент системного администрирования для решения различных задач, одной из которых является создание учетных записей для новых пользователей.

В других дистрибутивах, таких как SUSE Linux, Red Hat Linux или Mandriva, установка и система администрирования интегрированы в одной программе (такой, как *yast* или *yast2* в SUSE Linux).

Если нет другого способа, можно создать учетную запись вручную. Обычно для создания учетной записи необходимо сделать следующее:

1. Отредактировать файл */etc/passwd*, чтобы добавить нового пользователя. (Если воспользоваться для этого программой *vipw*, а не редактировать файл напрямую, можно застраховаться от параллельного изменения файла паролей, но *vipw* есть не во всех дистрибутивах.)
2. При необходимости отредактировать файл */etc/shadow* для задания теневого пароля (shadow password) нового пользователя.
3. Создать рабочий каталог пользователя.
4. Скопировать в рабочий каталог нового пользователя шаблоны файлов конфигурации (например, *.bashrc*). Иногда они находятся в каталоге */etc/skel*.

Мы не будем здесь вдаваться в особые детали: подробности того, как создаются новые учетные записи, можно найти практически в каждой книге по системному администрированию UNIX. О создании пользователей мы еще поговорим в разделе «Управление учетными записями пользователей» главы 11. Вы наверняка найдете утилиту, которая позаботится за вас обо всех деталях.

Запомните, что для установки или изменения пароля новой учетной записи используется команда `passwd`. Например, чтобы изменить пароль пользователя *duck*, выполните следующую команду:

```
# passwd duck
```

В результате вы получите приглашение установить или изменить пароль для *duck*. Если вы выполните команду `passwd`, будучи зарегистрированным как *root*, вам не будет предложено ввести старый пароль. Благодаря этому, если вы забыли свой пароль, но можете зарегистрироваться как *root*, вы сможете установить пароль заново.

Получение оперативной помощи

Linux предоставляет оперативную помощь в виде электронных страниц справочного руководства, или manual pages. На протяжении всей книги мы будем отправлять вас к страницам справочного руководства за получением дополнительных сведений об отдельных командах. Страницы руководства подробно описывают имеющиеся в системе программы и приложения, и важно знать, как получить доступ к этой оперативной документации в случае затруднений.

Для получения оперативной помощи по конкретной команде нужно использовать команду `man`. Например, для получения помощи по команде `passwd` введите следующее:

```
$ man passwd
```

В результате должна быть показана страница руководства для `passwd`.

Обычно страницы руководства поставляются как дополнительный пакет в дистрибутиве. Чтобы вы могли получить к ним доступ, этот пакет должен быть установлен. Мы настоятельно рекомендуем установить страницы руководства, потому что без их помощи вы слишком часто будете заходить в тупик.

Надо сказать, что некоторые страницы руководства могут быть неполными или отсутствовать в вашей системе. Все зависит от того, насколько полный у вас дистрибутив и насколько новое руководство.

На страницах руководства Linux описаны также системные вызовы, библиотечные функции, форматы файлов конфигурации и структура ядра. В разделе «Страницы справочного руководства» главы 4 более подробно описывается их использование.

Помимо обычных страниц руководства есть так называемые *инфо-страницы* (*Info pages*). Прочесть их можно с помощью текстового редактора Emacs, команды `info` или какого-либо из многочисленных графических средств чтения страниц Info.

Во многих дистрибутивах есть также документация в формате HTML, которую можно читать любым броузером типа Konqueror или редактором Emacs.

Наконец, есть файлы документации в простом текстовом формате. Их можно читать с помощью любого текстового редактора или просто командой `more`.

Если не удается найти документацию по некоторой команде, можно попробовать выполнить ее с параметром `-h` или `-help`, в этом случае большая часть команд кратко сообщат о том, как ими пользоваться.

Редактирование /etc/fstab

Чтобы обеспечить доступность всех файловых систем Linux после перезагрузки, может потребоваться отредактировать файл */etc/fstab*, описывающий файловые системы на машине. Многие дистрибутивы автоматически создают файл */etc/fstab* во время установки, тогда все просто. Однако, если у вас есть другие файловые системы, которые не использовались во время установки, может потребоваться добавить их в */etc/fstab* для того, чтобы они стали доступны. В */etc/fstab* должны быть также включены разделы для свопинга.

Для того чтобы иметь доступ к файловой системе, ее нужно *смонтировать* в системе. Монтирование файловой системы связывает ее с определенным каталогом. Например, корневая файловая система монтируется в каталог /, файловая система */usr* – в */usr*, и т. д. (Если вы не создали отдельной файловой системы для */usr*, все файлы из каталога */usr* будут храниться в корневой файловой системе.)

Мы не хотим сейчас перегружать вас техническими подробностями, но, прежде чем начать исследовать систему, необходимо понять, как сделать доступными файловые системы. Более подробно о монтировании файловых систем см. в разделе «Монтирование файловых систем» главы 10 или в любой книге по системному администрированию UNIX.

Корневая файловая система автоматически монтируется в каталог / при загрузке Linux. Однако остальные файловые системы надо монтировать отдельно. Обычно это делается командой

```
# mount -av
```

Ее помещают в один из стартовых файлов системы в */etc/rc.d* или другом месте, куда ваш дистрибутив записывает файлы конфигурации. Эта команда `mount`[1] монтирует все файловые системы, перечисленные в */etc/fstab*, поэтому для автоматического монтирования файловых систем во время загрузки необходимо включить их в */etc/fstab*. (Конечно, всегда можно вручную монтировать файловые системы после загрузки командой `mount`, но это лишняя работа.)

Ниже приведен пример файла */etc/fstab*. Он сокращен за счет того, что в каждой строке опущены два последних параметра, которые не являются обязательными и не относятся к нашему обсуждению. В данном примере корневая файловая система находится в разделе */dev/hda1*, файловая система */home* – в разделе */dev/hdb2*, а раздел для свопинга – на устройстве */dev/hdb1*:

```
# /etc/fstab
# device   directory   type   options
#
/dev/hda1  /           ext3   defaults
/dev/hdb2  /home       ext3   defaults
/dev/hdb1  none        swap   sw
/proc      /proc       proc   defaults
```

Строки, начинающиеся с символа `#`, являются комментариями. Обратите также внимание на дополнительную строку для файловой системы */proc*. Файловая система */proc* является виртуальной файловой системой, используемой некоторыми командами, такими как `ps`, для сбора данных о процессах.

Как вы видите, */etc/fstab* состоит из ряда строк. Первое поле каждой строки – имя устройства, например */dev/hda1*. Второе поле – точка монтирования, то есть каталог, в котором монтируется файловая система. Третье поле – тип файловой системы. Для файловых систем Linux *ext3fs* в этом поле нужно указать `ext3`, для разделов свопинга – `swap`. В четвертом поле указываются параметры монтирования. Для файловых систем в нем нужно указывать `defaults`, а для свопинга – `sw`.

Используя этот пример в качестве образца, вы сможете добавить строки для любых файловых систем, которых еще нет в файле */etc/fstab*.

Как добавлять строки в файл? Проще всего, зарегистрировавшись как *root*, отредактировать его с помощью редактора *vi* или Emacs. Сейчас мы не станем касаться использования текстовых редакторов – оба эти редактора описываются в главе 19.

После редактирования файла нужно выполнить команду

```
# /bin/mount -a
```

или перезагрузиться, чтобы изменения возымели действие.

Если вы завязли в этом месте, не тревожьтесь. Мы рекомендуем начинающим пользователям UNIX немного почитать об основах использования этой операционной системы и системном администрировании. В оставшейся части книги мы предполагаем знакомство читателя с этими основами, поэтому не говорите, что мы вас не предупреждали.

---

[1] Если совсем точно: команда `mount` с именно так записанными параметрами... – *Примеч. науч. ред.*

Останов системы

Никогда нельзя перезагружать или останавливать систему путем нажатия кнопки сброса (reset) или выключения питания. Как и большинство UNIX-систем, Linux кэширует в памяти запись на диск. Поэтому при внезапной перезагрузке системы без предварительного корректного останова можно повредить данные на дисках. Заметьте, что «комбинация из трех пальцев» (одновременное нажатие Ctrl+Alt+Delete) обычно безопасна: ядро перехватывает последовательность нажатий клавиш и инициирует корректный останов системы (как описано в разделе «Файлы init, inittab и rc» главы 17). В настройках вашей системы комбинация Ctrl+Alt+Delete может быть зарезервирована для использования только системным администратором, чтобы обычные пользователи не могли остановить сетевой сервер, от которого зависит работа целого подразделения. Чтобы ограничить использование этой комбинации клавиш, создайте файл */etc/shutdown.allow* и перечислите в нем имена всех пользователей, которым разрешается останавливать машину.

Проще всего остановить систему с помощью команды *shutdown*. Например, для немедленного останова и перезагрузки системы выполните, зарегистрировавшись как *root*, команду

```
# shutdown  -r now
```

Тем самым система будет корректно перезагружена. На странице руководства для *shutdown* описаны другие аргументы командной строки. Вместо `now` можно задать время, когда система должна быть перезагружена. В большинстве дистрибутивов есть также команда *halt*, вызывающая *shutdown now*. В некоторых дистрибутивах есть еще команда *poweroff*, которая останавливает компьютер и выключает его питание. Будет ли она работать, зависит не от Linux, а от аппаратной части и версии BIOS (которые должна поддерживать APM или ACPI).

Устранение неполадок

При первой попытке установить Linux почти все сталкиваются с какими-либо препятствиями. В основном проблемы возникают просто от недопонимания. Однако иногда могут быть более серьезные ситуации – промах разработчиков или ошибка в программе.

В этом разделе описаны некоторые наиболее часто возникающие проблемы и способы их решения. Здесь также говорится о некоторых неожиданных сообщениях об ошибках, появляющихся во время установки, которая выглядит успешной.

Нормальная последовательность загрузки обычно следующая:

1. После загрузки с приглашения LILO система должна загрузить образ ядра с дискеты. Для этого может понадобиться несколько секунд; процесс идет нормально, если лампочка привода гибких дисков продолжает светиться.
2. Во время загрузки ядра должны быть проверены устройства SCSI. Если у вас нет устройств SCSI, система «зависнет» примерно на 15 секунд, пока производится поиск устройств SCSI; обычно это происходит после вывода строки

```
lp_init: lp1 exists (0), using polling driver
```

3. По окончании загрузки ядра управление передается загрузочным файлам системы на гибком диске. В конце будет выведено приглашение регистрации или запущена программа установки. Если показано приглашение регистрации в виде:

```
Linux login:
```

нужно зарегистрироваться (обычно как *root* или *install*, что зависит от дистрибутива, который вы используете). После ввода имени пользователя система может замолчать секунд на 20 или более, пока с дискеты загружается программа установки или оболочка. При этом должна гореть лампочка привода гибких дисков. Не думайте, что система зависла.

Проблемы с загрузочным и установочным носителем

При первой попытке загрузки с установочного носителя можно столкнуться с рядом проблем. Отметим, что описываемые проблемы не относятся к загрузке только что установленной системы Linux. Об этих проблемах читайте в разделе «Проблемы, возникающие после установки Linux».

При попытке загрузки возникает ошибка обращения к дискете или носителю

Самой частой причиной такой ошибки является повреждение загрузочной дискеты. Либо дискета повреждена физически, и тогда вам нужно создать новый загрузочный диск на другой дискете, либо испорчены данные на дискете, и тогда нужно проверить правильность загрузки данных и копирования их на дискету. Во многих случаях проблема решается повторным созданием загрузочной дискеты. Повторите операции по созданию дискеты и попробуйте загрузиться еще раз.

Если вы получили загрузочную дискету по почте или у какого-либо дистрибьютора, а не загружали данные из Интернета и не создавали дискету сами, попросите дистрибьютора предоставить вам новую загрузочную дискету, но только убедившись сначала, что проблема состоит именно в дискете. Конечно, это может быть нелегко, но если привод гибких дисков издает необычные звуки или появляются сообщения типа «невозможно прочесть сектор», весьма вероятно, что поврежден носитель.

Машина зависает во время загрузки или после нее

После начала загрузки установочной дискеты ядро выводит ряд сообщений, указывающих на то, какие устройства были обнаружены и сконфигурированы. После этого обычно следует приглашение регистрации, позволяющее продолжить установку (некоторые дистрибутивы вместо этого сразу запускают программу установки). На этом этапе может показаться, что система зависла. Будьте терпеливы, загрузка программ с дискеты происходит очень медленно. Часто система вовсе не зависла, просто ей требуется время. Прежде чем решить, что система зависла, убедитесь, что активности дисков или компьютера не отмечается в течение нескольких минут.

Каждая из перечисленных в начале раздела операций может вызвать задержку, которая заставит предположить, что система встала. Однако система может и действительно зависнуть во время загрузки по ряду причин. Во-первых, в системе может оказаться недостаточно оперативной памяти для за-

грузки инсталляционного носителя. (См. в следующем пункте, как отключить электронный диск для освобождения оперативной памяти.)

Во-вторых, причиной зависания во многих случаях может быть несовместимость аппаратного обеспечения. Даже если имеющиеся аппаратные средства поддерживаются системой, могут возникнуть проблемы с несовместимостью аппаратных конфигураций, которые вызовут зависание системы. Ниже в разделе «Проблемы с аппаратным обеспечением» рассказывается об аппаратной несовместимости. В разделе «Требования к оборудованию» главы 16 перечислены поддерживаемые в настоящее время наборы микросхем видеосистемы, с которыми связаны основные проблемы работы Linux в графическом режиме.

При начальной загрузке или установке программного обеспечения система сообщает о нехватке памяти (out-of-memory)

Эта проблема связана с объемом оперативной памяти, установленной в машине. Самой системе Linux требуется для работы не менее 8 Мбайт памяти. Практически все современные дистрибутивы требуют 32 Мбайт или более. Если в машине 16 Мбайт памяти или меньше, могут возникнуть неприятности при загрузке установочного носителя или установке самого программного обеспечения. Это происходит из-за того, что во многих дистрибутивах используется *электронный диск* (*ramdisk*), который представляет собой файловую систему, загружаемую непосредственно в оперативную память и используемую для действий с установочным носителем. Например, в электронный диск может быть загружен весь образ установочной дискеты, для чего нужно больше 1 Мбайт памяти.

Решение проблемы состоит в том, чтобы отключить возможность использования электронного диска на время загрузки установочного носителя. В каждом дистрибутиве для этого существует своя процедура. Сведения о ней должны быть в документации, сопровождающей дистрибутив.

Сообщения об ошибке «out-of-memory» во время загрузки и установки можно и не увидеть. Система просто неожиданно зависнет или прекратит загрузку. Если система зависла по причинам, не связанным с объяснениями предыдущего раздела, попробуйте отключить электронный диск.

Во время загрузки система сообщает об ошибке, например «Permission denied» (В доступе отказано) или «File not found» (Файл не найден)

Это свидетельствует о повреждении инсталляционного носителя. При загрузке с инсталляционного носителя (если вы все делаете правильно) таких сообщений быть не должно. Свяжитесь со своим дистрибьютором программного обеспечения Linux и сообщите о проблеме; при необходимости получите другую копию загрузочного носителя. Если вы загрузили диск из Интернета сами, попробуйте его создать заново, возможно, это решит ваши проблемы.

При загрузке система сообщает об ошибке: «VFS: Unable to mount root»

Это сообщение означает, что корневая файловая система (находящаяся на самом загрузочном носителе) не найдена, и свидетельствует о том, что ваш загрузочный носитель поврежден или вы неправильно загружаете систему.

Например, многие дистрибутивы на CD-ROM/DVD требуют, чтобы во время загрузки компакт-диск или DVD-диск находился в приводе. Проверьте также, включен ли привод CD-ROM/DVD, и есть ли к нему обращения. Возмож-

но, система не может найти привод CD-ROM/DVD во время загрузки. Более подробные сведения вы найдете в следующем разделе.

Если вы уверены, что правильно производите загрузку, возможно, ваш загрузочный носитель действительно поврежден. Это происходит редко, поэтому попробуйте найти другие решения, прежде чем пытаться использовать другой загрузочный диск или ленту. В дистрибутиве Red Hat есть полезная новая функция `mediacheck`, с помощью которой можно проверить целостность компакт-диска. Она позволит убедиться, что с носителем все в порядке.

Проблемы с аппаратным обеспечением

Наиболее часто встречающаяся проблема при установке или работе с Linux – несовместимость аппаратного обеспечения. Даже если все ваше аппаратное обеспечение поддерживается Linux, его неправильная настройка или конфликт аппаратных устройств могут приводить к необычным результатам: устройства не определяются при загрузке или система виснет.

Важно идентифицировать подобные проблемы с аппаратурой, если есть подозрение, что они могут быть причиной возникших проблем. В следующих разделах мы опишем некоторые распространенные проблемы с аппаратным обеспечением и способы их решения.

Идентификация проблем, связанных с аппаратным обеспечением

Если возникла проблема, которая предположительно связана с аппаратным обеспечением, для начала ее надо идентифицировать. Это значит, что нужно исключить все возможные переменные составляющие и, как часто бывает, разобрать систему по кусочкам, пока не удастся выделить элемент аппаратуры, вызывающий проблемы.

Все не так страшно, как может показаться. Обычно нужно убрать из машины все несущественные устройства (предварительно выключив машину из сети), а затем определить, какое из устройств вызывает проблемы – возможно, путем поочередной установки каждого устройства. Это означает, что нужно убрать все устройства, кроме контроллера гибкого диска, видеоадаптера и, конечно, клавиатуры. Даже такое невинно выглядящее устройство, как контроллер мыши, может оказаться нарушителем спокойствия, если вы сочтете его малозначительным. Поэтому на время проведения экспериментов на всякий случай удалите все, что не является абсолютно необходимым для начальной загрузки, и собирайте потом машину заново, добавляя устройства по одному.

Допустим, что система зависает во время загрузки при обнаружении платы Ethernet. Вы предполагаете, что существует конфликт или проблема с картой Ethernet, установленной в вашей машине. Простой и быстрый способ выяснить это состоит в том, чтобы вынуть карту Ethernet и попытаться снова загрузиться. Если это удастся, то либо ваша карта Ethernet не поддерживается Linux, либо она вызывает конфликт по адресам ввода-вывода или по номеру IRQ. Кроме того, некоторые скверные сетевые карты (в основном это относится к клонам NE2000 на шине ISA, которые, к счастью, сейчас вымирают) могут подвесить всю систему во время автоматической проверки. В таком случае лучше всего вынуть сетевую карту на время установки и вернуть ее на место позднее либо при загрузке пере-

дать ядру необходимые параметры, чтобы избежать автоматической проверки сетевой карты. Самое верное средство – избавиться от этой карты и приобрести новую у производителя, который более тщательно разрабатывает свои изделия.

Вы можете спросить, что означает «конфликт адресов ввода-вывода или номер IRQ»? Все устройства в вашей машине используют линию запроса прерывания, или IRQ, с помощью которой сообщают системе о том, что она должна что-то для них сделать. Можете представлять себе IRQ как веревочку, за которую дергает устройство, когда ему нужно, чтобы система выполнила какое-нибудь возникшее у него требование. Если за одну и ту же веревочку дергают несколько устройств, система не может определить, которое из устройств она должна обслужить. Мгновенно возникает каша.

Поэтому обеспечьте использование всеми установленными у вас устройствами, если они не на шине PCI/AGP, уникальных линий IRQ. Обычно IRQ для устройства можно установить перемычками на карте. Как это сделать, должно быть описано в документации по конкретному устройству. Для некоторых устройств IRQ вообще не требуется, но вам предлагается сконфигурировать их для использования некоторого IRQ, если это возможно (примерами таких устройств являются SCSI-контроллеры Seagate ST01 и ST02). Шина PCI спроектирована более удачно, и подключаемые к ней устройства могут использовать линии прерывания совместно.

В некоторых случаях ядро, находящееся на установочном носителе, сконфигурировано для использования определенных IRQ с определенными устройствами. Например, в некоторых дистрибутивах Linux ядро заранее настроено на использование IRQ 5 для SCSI-контроллера TMC-950, контроллера CD-ROM Mitsumi и драйвера шинной мыши. Если нужно использовать два или более из этих устройств, следует сначала установить Linux с одним из них, а затем перекомпилировать ядро, чтобы изменить значение IRQ по умолчанию одного из этих устройств. (О перекомпиляции ядра см. раздел «Сборка нового ядра» в главе 18.)

Другими областями, где могут возникать аппаратные конфликты, являются каналы прямого доступа к памяти (DMA), адреса ввода-вывода и адреса совместно используемой памяти. Все эти термины описывают механизмы взаимодействия системы с аппаратными устройствами. Например, некоторые карты Ethernet используют для взаимодействия с системой как адреса совместно используемой памяти, так и IRQ. Если они вступают в конфликт с другими устройствами, система может вести себя непредсказуемо. Вы должны с помощью перемычек изменить у этих устройств канал DMA, адреса ввода-вывода и совместно используемой памяти. (К несчастью, некоторые устройства не позволяют этого сделать.)

В документации к вашим аппаратным устройствам должны быть указаны IRQ, канал DMA, адреса ввода-вывода и совместно используемой памяти, используемые устройством, а также способ их настройки. Проблема, разумеется, в том, что некоторые из этих установок нельзя узнать, не собрав систему, а потому их может не быть в документации. Простой способ обойти эти проблемы все тот же – временно отключить конфликтующие устройства и разобраться с причиной возникшей проблемы позднее.

В табл. 2.1 приведен список каналов IRQ и DMA, используемых различными стандартными устройствами в большинстве компьютеров. Эти устройства есть

почти во всех машинах, поэтому избегайте установки тех же IRQ или DMA в других устройствах.

Таблица 2.1. Распространенные настройки устройств

Устройство	Адрес I/O	IRQ	DMA
ttyS0 (COM1)	3f8	4	Не используется
ttyS1 (COM2)	2f8	3	Не используется
ttyS2 (COM3)	3e8	4	Не используется
ttyS3 (COM4)	2e8	3	Не используется
lp0 (LPT0)	378–37f	7	Не используется
lp1 (LPT2)	278–27f	5	Не используется
fd0, *fd1* (гибкие диски 1 и 2)	3f0–3f7	6	2
fd2, *fd3* (гибкие диски 3 и 4)	370–377	10	3

Проблемы при распознавании жесткого диска или контроллера

В начале загрузки Linux вы видите на экране ряд сообщений, например:

```
Console: switching to colour frame buffer device 147x55
Real Time Clock Driver v1.12
Serial: 8250/16550 driver $Revision: 1.7 $ 48 ports, IRQ sharing enabled
ttyS0 at I/O 0x3f8 (irq = 4) is a 16550A
ttyS1 at I/O 0x2f8 (irq = 3) is a 16550A
Using anticipatory io scheduler
Floppy drive(s): fd0 is 1.44M
FDC 0 is a post-1991 82077
...
```

Здесь ядро сообщает об обнаружении в вашей машине различных аппаратных устройств. В некоторый момент должна появиться строка, содержащая примерно следующее:

```
hda: hda1 hda2 hda3 hda4 < hda5 hda6 hda7 >
```

Если по какой-либо причине ваши диски или разделы окажутся нераспознанными, вы никак не сможете ими воспользоваться.

Это может произойти в нескольких ситуациях:

Жесткий диск или контроллер не поддерживаются

Если ваш контроллер жесткого диска (IDE, SCSI или иной) не поддерживается Linux, то ядро не сможет распознать разделы во время загрузки.

Диск или контроллер неправильно сконфигурированы

Даже если Linux поддерживает ваш контроллер, он может оказаться неверно настроенным. (Обычно эта проблема возникает со SCSI-контроллерами, большая часть других контроллеров должна работать без дополнительной настройки.)

Для решения такого рода проблем обратитесь к документации по вашему жесткому диску и контроллеру. В частности, на многих жестких дисках нужно

установить перемычку, если он будет использоваться как slave, то есть в качестве второго жесткого диска. Проверкой в такой ситуации будет загрузка Windows или другой операционной системы, относительно которой известно, что она может работать с вашим диском и контроллером. Если диск и контроллер доступны из другой операционной системы, то проблема не в аппаратной конфигурации.

Почитайте о разрешении возможных конфликтов между устройствами в предыдущем разделе, «Идентификация проблем, связанных с аппаратным обеспечением», и о конфигурировании SCSI-устройств – в следующем разделе, «Решение проблем, возникающих со SCSI-контроллерами и SCSI-устройствами».

Контроллер сконфигурирован правильно, но не обнаруживается

Некоторые SCSI-контроллеры без BIOS требуют, чтобы пользователь ввел сведения о контроллере во время загрузки. В следующем разделе, «Решение проблем, возникающих со SCSI-контроллерами и SCSI-устройствами», описано, как обеспечить обнаружение аппаратных устройств для таких контроллеров.

Не распознается геометрия диска

В некоторых системах, таких как IBM PS/ValuePoint, данные о геометрии не записываются в память CMOS, где Linux рассчитывает их обнаружить. Кроме того, некоторые SCSI-контроллеры требуют, чтобы им указали, где находятся данные о геометрии диска, и тогда Linux сможет распознать структуру вашего диска.

В большинстве дистрибутивов есть параметр загрузки, позволяющий задать геометрию диска. Обычно при загрузке с инсталляционного носителя можно задать геометрию диска в ответ на приглашение загрузки LILO командой:

```
boot: linux hd=cylinders,heads,sectors
```

где параметры *cylinders*, *heads* и *sectors* определяют число цилиндров, головок и секторов, соответственно, на дорожке вашего жесткого диска.

После установки программного обеспечения можно установить LILO, что позволит загружать систему с жесткого диска. В этот момент можно задать геометрию диска в процедуре установки LILO, что избавит от необходимости вводить геометрию при каждой загрузке. Более подробно о LILO см. в разделе «Использование GRUB» главы 17.

Решение проблем, возникающих со SCSI-контроллерами и SCSI-устройствами

Ниже описываются некоторые наиболее распространенные проблемы, возникающие со SCSI-контроллерами и SCSI-устройствами, такими как CD-ROM, жесткие диски и ленточные приводы. Если возникли проблемы с распознаванием Linux диска или контроллера, прочтите этот раздел. Еще раз отметим, что большинство дистрибутивов используют модульное ядро, и, возможно, вам было необходимо загрузить модуль, поддерживающий ваше устройство, на ранней стадии процесса установки. Это может выполняться и автоматически.

В Linux «SCSI HOWTO» содержится много полезных дополнительных сведений о SCSI-устройствах. SCSI-устройства иногда конфигурируются весьма замысловатым образом.

Например, использование дешевых шлейфов может оказаться ложной экономией, особенно для wide SCSI. Дешевые кабели – основной источник проблем; они могут вызывать все виды отказов. Если у вас есть устройства SCSI, используйте надлежащие кабели.

Вот наиболее часто встречающиеся проблемы и их возможные решения:

Устройство SCSI определяется на всех возможных ID

Эта проблема возникает, когда система назначает устройству тот же адрес, который имеет контроллер. Необходимо изменить положение перемычек, чтобы диск использовал адрес, отличный от адреса контроллера.

Linux сообщает об отсутствии доступа, хотя известно, что устройства исправны

Это может быть вызвано плохими шлейфами или терминаторами. Если шина SCSI не имеет терминаторов на обоих концах, то при доступе к устройствам SCSI могут возникать ошибки. Если у вас есть сомнения, обязательно проверьте шлейфы. Нередко источником ошибок являются неприсоединенные или имеющие низкое качество шлейфы.

Устройства SCSI сообщают об ошибках тайм-аута

Обычно это вызвано конфликтом IRQ, DMA или адресов устройств. Проверьте также правильность установки прерываний контроллера.

Не определяются SCSI-контроллеры с использованием BIOS

Обнаружение контроллеров с использованием BIOS, может быть безуспешным, если BIOS отключена или сигнатура контроллера не распознается ядром. Более подробно читайте об этом в Linux «SCSI HOWTO».

Не работают контроллеры, использующие отображаемый в памяти ввод-вывод

Это происходит, когда отображаемые в памяти порты ввода-вывода неправильно кэшируются. Пометьте в XCMOS адресное пространство платы как некэшируемое либо вообще отключите кэш.

При создании разделов выводится предупреждение «cylinders>1024» или невозможно загрузить систему с раздела, использующего цилиндры с номерами, большими 1023

BIOS ограничивает количество цилиндров числом 1024, и любой раздел, использующий цилиндры с номерами, бо**Ъ**шими этого числа, недоступен через BIOS. Что касается Linux, это оказывает влияние только на загрузку. Если система загрузилась, вы должны получить доступ к такому разделу. Решение – загружать Linux либо с загрузочной дискеты, либо с раздела, использующего цилиндры до номера 1024. См. раздел «Создание загрузочной дискеты или установка GRUB» выше в этой главе.

Во время загрузки не распознается привод CD-ROM или другого съемного носителя

Попробуйте загрузиться при находящемся в приводе компакт-диске (или дискете). Для некоторых устройств это необходимо.

Если не распознается SCSI-контроллер, может потребоваться осуществить принудительное определение устройства во время загрузки. Это особенно важно для SCSI-контроллеров без BIOS. Большинство дистрибутивов позволяют задать IRQ контроллера и адрес совместно используемой памяти при за-

грузке с установочного носителя. Например, при использовании контроллера TMC-8xx можно в ответ на приглашение загрузки LILO ввести:

```
boot: inux tmx8xx=interrupt,memory-address
```

где *interrupt* — это IRQ контроллера, а *memory-address* — адрес совместно используемой памяти. Наличие такой возможности зависит от того, какой дистрибутив вами используется, поэтому изучите поставляемую с дистрибутивом документацию.

Проблемы при установке программного обеспечения

Если вам повезет, то установка программного обеспечения пройдет без помех. Проблемы, с которыми вы можете столкнуться, могут быть связаны с повреждением инсталляционного носителя или недостатком места в файловых системах Linux. Вот перечень наиболее часто встречающихся проблем:

Во время установки программного обеспечения появляется сообщение «Read error, file not found» или другое сообщение об ошибке

Это указывает на проблемы с инсталляционным носителем. При установке с дискет имейте в виду, что дискеты весьма подвержены такого рода ошибкам носителя. Используйте новые, заново отформатированные дискеты. Если на вашем диске есть раздел Windows, то можно воспользоваться тем, что многие дистрибутивы Linux позволяют устанавливать программное обеспечение с жесткого диска. Такой подход может оказаться более быстрым и надежным, чем установка с дискет.

При установке с компакт-диска убедитесь в отсутствии царапин, пыли или других помех, способных вызвать ошибки носителя.

Причиной проблемы может также быть неверный формат носителя. Например, при использовании дискет многие дистрибутивы Linux требуют, чтобы дискеты были отформатированы в формате Windows с высокой плотностью записи. (Исключением является загрузочная дискета: в большинстве случаев она имеет формат, отличный от Windows.) Если ничто другое не помогает, получите новый комплект дискет или заново их создайте, используя новые дискеты, если вы самостоятельно загрузили программное обеспечение из сети.

Система выводит сообщение «tar: read error» или «gzip: not in gzip format»

Обычно эти ошибки связаны с искажением файлов на инсталляционном носителе. Иными словами, сама дискета цела, но данные на ней каким-то образом искажены. Например, если вы загрузили программное обеспечение в текстовом, а не в двоичном режиме, то ваши файлы окажутся искажены и не смогут быть прочтены программой установки. При загрузке по FTP выполните команду *binary*, чтобы установить правильный режим, прежде чем запрашивать передачу файлов.

Во время установки система выводит сообщение «device full» (нет места на устройстве) или аналогичное

Это четкое указание на то, что у вас недостаточно места для установки программного обеспечения. При переполнении диска не все дистрибутивы могут корректно восстановить работу, и в результате процесса установки вы не получите работающей системы.

Решение проблемы обычно состоит в повторном создании файловых систем командой *mke2fs*, которая удалит частично установленное программное обеспечение. Затем можно попытаться еще раз установить программное обеспечение, на этот раз отобрав для установки меньше программ. Если же вам совершенно необходимы эти программы, то может потребоваться запустить установку с самого начала и переопределить размеры разделов и файловых систем.

При обращении к жесткому диску система выводит сообщение типа «read_intr: 0x10»

Обычно это свидетельствует о наличии на жестком диске сбойных блоков. Однако, если такие сообщения возникают во время работы утилит *mkswap* или *mke2fs*, может оказаться, что системе не удается обратиться к жесткому диску из-за ошибок устройства (см. раздел «Проблемы с аппаратным обеспечением» ранее в этой главе) или неправильно указанной геометрии диска. Если при загрузке используется параметр:

```
hd=cylinders,heads,sectors
```

для принудительного определения геометрии диска и он задан неправильно, то можно получить такое сообщение. То же самое может произойти при неправильном задании геометрии диска в системном CMOS.

Система выводит сообщение «file not found» (файл не найден) или «permission denied» (доступ запрещен)

Это может произойти, если на инсталляционном носителе присутствуют не все файлы или программа установки столкнулась с проблемами прав доступа. Известно, например, что в некоторых дистрибутивах Linux есть ошибки в самой программе установки. Такие ошибки быстро исправляются и встречаются довольно редко. Если вы подозреваете, что в программном обеспечении дистрибутива есть ошибки, и уверены, что все делали правильно, свяжитесь с тем, кто осуществляет сопровождение дистрибутива, и сообщите об ошибке.

Если во время установки Linux возникают другие непонятные ошибки (особенно если вы самостоятельно загрузили программное обеспечение), проверьте, действительно ли вы загрузили все необходимые файлы.

Например, некоторые используют для загрузки программного обеспечения по FTP команду:

```
mget *.*
```

В результате загружаются только файлы, в именах которых содержится точка, а если есть файлы без точки, то они не будут получены. Правильная команда должна быть следующей:

```
mget *
```

Лучше всего в случае неприятностей повторить все действия заново. Вам может казаться, что вы все сделали правильно, а в действительности где-то по дороге вы пропустили маленький, но важный шаг. Во многих случаях достаточно заново загрузить или снова установить программное обеспечение, чтобы решить проблему. Не стоит слишком долго биться головой о стену!

Кроме того, если Linux неожиданно зависает во время установки, это может быть результатом тех или иных проблем с аппаратурой. Соответствующие советы см. в разделе «Проблемы с аппаратным обеспечением».

Проблемы, возникающие после установки Linux

Вы провели целый день за установкой Linux. Чтобы освободить для нее место, вы затерли разделы Windows и OS/2 и со слезами удалили копию SimCity 2000 и Railroad Tycoon 2. Вы перезагружаете систему, и ничего не происходит. Или, еще хуже, что-то происходит, но не то, что должно произойти. Что делать?

В разделе «Проблемы с загрузочным и установочным носителем» этой главы мы рассказали о наиболее часто встречающихся проблемах при загрузке инсталляционного носителя Linux. Описанные причины могут действовать и в данной ситуации. Кроме того, вы могли стать жертвой следующих напастей.

Проблемы при загрузке Linux с дискеты

При загрузке Linux с дискеты может потребоваться задать расположение корневого раздела Linux. Это особенно касается случая, когда для загрузки используется сама установочная дискета, а не специальная загрузочная дискета, созданная во время установки.

При загрузке с дискеты удерживайте нажатой клавишу Shift или Ctrl. Это должно привести к появлению меню загрузки. Нажмите клавишу Tab, чтобы вывести список возможных вариантов. Например, многие дистрибутивы позволяют загрузиться с дискеты после ввода следующей строки:

```
boot: linux root=partition
```

где *partition* – имя корневого раздела Linux, например */dev/hda2*. Дистрибутив SUSE Linux предлагает меню на ранней стадии выполнения установочной программы, которая загружает вновь созданную вами систему Linux с установочной загрузочной дискеты. Подробности смотрите в документации дистрибутива.

Проблемы при загрузке Linux с жесткого диска

Если вы решили установить LILO, а не создавать загрузочную дискету, у вас должна быть возможность загрузить Linux с жесткого диска. Однако используемая во многих дистрибутивах автоматизированная процедура установки LILO не всегда совершенна. Она может сделать неправильные предположения о структуре вашего раздела, и тогда вам понадобится переустановить LILO, чтобы все заработало. Установка загрузчика LILO описана в разделе «Использование GRUB» главы 17.

Вот некоторые часто возникающие проблемы:

Система выводит сообщение «Drive not bootable – Please insert system disk» (Диск не загрузочный – вставьте системный диск)

Это сообщение об ошибке появляется при повреждении главной загрузочной записи жесткого диска. В большинстве случаев это не опасно, и все остальные данные на диске целы. Есть несколько способов справиться с этой проблемой:

- При разбиении диска на разделы с помощью *fdisk* вы могли удалить раздел, помеченный как active (активный). Windows и другие операционные

системы пытаются осуществлять загрузку с активного раздела (Linux обычно не обращает внимания, является ли раздел активным, но некоторые дистрибутивы, например Debian, устанавливают главную загрузочную запись, которая ищет активный раздел). Можно загрузить MS-DOS с дискеты и, запустив *fdisk*, установить флаг активности на разделе MS-DOS, и тогда все пойдет нормально.

Другая команда, которую можно использовать (с MS-DOS 5.0 и выше, а также в Windows начиная с Windows 95):

```
FDISK /MBR
```

Эта команда попытается переписать главную загрузочную запись жесткого диска для загрузки Windows, затерев LILO. Если Windows на жестком диске у вас не осталась, нужно загрузить Linux с дискеты и попробовать установить LILO позже. Данная команда отсутствует в Windows NT/2000/XP, поэтому процедура восстановления загрузочной записи Windows в этих операционных системах более сложная.

- Такую ошибку можно получить, если создать раздел Windows с помощью Linux-версии *fdisk* или наоборот. Разделы Windows можно создавать только с помощью версии *fdisk* для Windows. (То же относится и к операционным системам, отличным от Windows.) Самое лучшее решение – начать все с самого начала и правильно разбить диск на разделы либо просто удалить плохие разделы и создать их заново с помощью корректной версии *fdisk*.
- Процедура установки LILO могла пройти неудачно. В этом случае нужно перезагрузиться с загрузочной дискеты Linux (если она есть) или с исходного инсталляционного носителя. В любом случае вы получите возможность указать во время загрузки корневой раздел Linux. Во время загрузки удерживайте нажатой клавишу Shift или Ctrl и нажмите Tab, находясь в загрузочном меню, чтобы вывести список возможных вариантов.

При загрузке системы с жесткого диска вместо Linux запускается Windows (или другая операционная система).

Сначала проверьте, действительно ли вы установили LILO во время установки программного обеспечения Linux. Если нет, то система будет по-прежнему загружать Windows (или другую имеющуюся операционную систему) при попытке загрузки с жесткого диска. Чтобы загружать с жесткого диска Linux, необходимо установить LILO или GRUB (см. раздел «Использование GRUB» главы 17).

Если же LILO действительно установлен, но вместо Linux загружается другая операционная система, значит, вы сконфигурировали LILO таким образом, что по умолчанию загружается другая операционная система. Во время загрузки системы удерживайте нажатой клавишу Shift или Ctrl и нажмите Tab, находясь в загрузочном меню. Вам должен быть выведен список операционных систем, которые можно загрузить. Для загрузки Linux выберите соответствующий пункт меню (обычно просто `linux`).

Если вы хотите, чтобы Linux стала операционной системой, загружаемой по умолчанию, необходимо перенастроить LILO.

Возможно также, что вы пытались установить LILO, но процедура установки по какой-то причине не завершилась. См. предыдущий пункт об установке.

Проблемы при входе в систему

После загрузки Linux должно выводиться приглашение для входа в систему:

```
Linux login:
```

Что делать дальше, говорится в документации дистрибутива или сообщает сама система. Во многих дистрибутивах вы регистрируетесь просто как *root* без пароля. Другими возможными именами пользователя могут быть *guest* или *test*.

Большинство дистрибутивов Linux запрашивают начальный пароль суперпользователя *root*. Надеемся, что вы запомнили пароль, который вводили при установке, – он вам теперь снова нужен. Если при установке дистрибутива пароль не запрашивался, можно попробовать ввести пустой пароль.

Если вы просто не можете зарегистрироваться, посмотрите документацию дистрибутива: необходимые имя пользователя и пароль должны быть в ней указаны. Имя пользователя и пароль могли быть даны вам во время процедуры установки или они могут быть выведены на баннере регистрации. Еще один способ заключается в том, чтобы загрузить Linux в однопользовательском режиме, для чего нужно в меню загрузки выбрать пункт `linux simple`.

Возможной причиной неудачи с паролем может быть проблема с установкой файлов регистрации и инициализации Linux. В этом случае может потребоваться переустановка программного обеспечения (или только какой-то его части) либо нужно загрузиться с инсталляционного носителя Linux и попытаться устранить проблему вручную.

Проблемы при работе с системой

Если регистрация прошла успешно, должно быть выдано приглашение оболочки (например, `#` или `$`), а возможно, автоматически будет загружено графическое окружение рабочего стола, такого как KDE или GNOME, после чего можно начать путешествие по системе. В этом случае следующим шагом должны стать процедуры, описанные в главе 4. Однако иногда всплывают некоторые начальные проблемы использования системы.

Самой распространенной проблемой начальной настройки являются некорректные права доступа к файлам или каталогам. В этом случае после регистрации может появиться сообщение об ошибке:

```
Shell-init: permission denied (доступ запрещен)
```

В действительности в любом случае, когда вы видите сообщение `permission denied`, можно быть уверенным, что проблема в правах доступа к файлам.

Часто достаточно воспользоваться командой `chmod`, чтобы исправить права доступа к соответствующим файлам или каталогам. Например, в некоторых дистрибутивах Linux когда-то использовались неправильные права доступа `0644` для корневого каталога (`/`). Для решения проблемы нужно, зарегистрировавшись как *root*, выполнить команду:

```
# chmod 755 /
```

(О правах доступа к файлам рассказано в разделе «Владение файлами и права доступа» главы 11.) Однако для того чтобы выполнить эту команду, необходимо загрузиться с инсталляционного носителя и смонтировать корневую файловую систему вручную – это сложная задача для большинства новичков.

Во время работы в системе вы вполне можете столкнуться с тем, что права доступа к некоторым файлам и каталогам некорректны или программное обеспечение работает не в соответствии с настройками. Добро пожаловать в мир Linux! Хотя большинство дистрибутивов вполне работоспособны, нельзя ожидать от них идеальной работы. Мы не будем рассказывать здесь обо всех проблемах, так как на протяжении всей книги мы будем помогать избавиться от многих из этих проблем настройки, показывая, как можно их обнаружить и решить самостоятельно. В главе 1 мы более или менее подробно обсуждали эту философию. Во второй части книги мы дадим советы по решению многих из часто встречающихся проблем настройки.

3

Окружение рабочего стола

Наиболее популярные современные дистрибутивы Linux обладают весьма привлекательным графическим интерфейсом. В этой главе рассказывается, как пользоваться этим интерфейсом для выполнения тех или иных действий быстро и с наименьшими усилиями. Большинство систем Linux предоставляют комплексную графическую среду, которая называется *рабочим столом* (*desktop*).

Эта глава охватывает два наиболее популярных рабочих стола Linux – K Desktop Environment (KDE) и GNOME. Те из читателей, кто испытывает затруднения в настройке графической среды или просто хочет получить более глубокие знания о графике в Linux, найдут в главе 16 ответы на многие свои вопросы.

Зачем нужен графический рабочий стол?

Если предполагается использовать Linux в качестве сервера, нет никакой необходимости устанавливать пакеты, которые будут описаны в этой главе (разве что если вы собираетесь использовать инструменты администрирования с графическим интерфейсом). X Window и окружение рабочего стола потребляют значительную долю памяти, процессорного времени и дискового пространства, поэтому, если к серверу даже не будет подключаться монитор, установка их будет лишь пустой тратой времени и ресурсов. Точно так же при разработке программного обеспечения, когда не предполагается разрабатывать программы с графическим интерфейсом или использовать интегрированные среды разработки (IDE), вполне можно обойтись без этих удобств.

Но во всех остальных случаях KDE и GNOME делают Linux привлекательнее для массового потребителя. Эти две графические среды предоставляют все, что только может потребоваться обычному среднему пользователю, например:

- Отображают содержимое файлов, когда пользователь щелкает мышью на их ярлыках, автоматически определяя тип файла и выбирая требуемую программу просмотра.
- Позволяют перемещать между окнами текст и графические изображения через буфер обмена, даже если эти окна принадлежат разным приложениям, которые хранят свои данные в совершенно разных форматах.

- Сохраняют и восстанавливают текущее состояние среды. Таким образом, когда пользователь опять входит в систему, он обнаруживает свой рабочий стол в том состоянии, в каком он его оставил.
- Помогают пользователю ориентироваться в обстановке с помощью ярлыков и всплывающих подсказок.
- Предлагают большое количество красивых «обоев» для рабочего стола, хранителей экрана и тем.
- Позволяют полностью перенастроить себя, но обладают такими настройками по умолчанию, которые прекрасно подходят большинству пользователей.

Для обеспечения всех этих особенностей и KDE, и GNOME требуют наличия достаточно быстрого микропроцессора и значительного объема памяти. Все современные компьютеры обладают достаточными вычислительными мощностями, чтобы обеспечить работу этих двух сред (а они с каждым годом становятся все привлекательнее), но некоторые пользователи предпочитают использовать более легковесные графические системы, несмотря на то, что они не обладают такой же мощностью. Если вы хотите получить нечто среднее между простой текстовой консолью и ресурсоемкими KDE или GNOME, попробуйте пользоваться оконным менеджером xfce. Он входит в состав многих дистрибутивов и может быть загружен с сайта *http://www.xfce.org* вместе с документацией. Намного более легкий, чем KDE или GNOME, он обладает на удивление богатым набором функциональных возможностей.

KDE и GNOME создавались так, чтобы быть максимально интуитивно понятными, и потому заимствовали множество интересных идей из других популярных графических оболочек, что сделало их простыми и понятными в использовании для большинства пользователей. В этой главе мы рассмотрим основные приложения и некоторые интересные элементы этих двух рабочих столов, которые вы могли не заметить в повседневной работе.

K Desktop Environment

K Desktop Environment – это проект open source, призванный предоставить удобный, полнофункциональный, современный и дружественный рабочий стол для UNIX и, следовательно, для Linux. Со времени начала проекта в октябре 1996 года были достигнуты значительные успехи. Частично этот прогресс обязан выбору очень высококачественного средства разработки Qt[1], а также последовательному использованию C++ и его объектно-ориентированных возможностей.

Рабочий стол KDE основан на KParts – компонентной технологии, которая в числе прочего позволяет встраивать одно приложение в другое, например, веб-бро-

[1] Лицензионные ограничения Qt, на которой основана программа KDE, – самое слабое место этого продукта в сравнении с GNOME, развиваемой полностью в рамках лицензии GNU GPL; в конце 1998 года разработчики Qt объявили о смягчении своей лицензии, но она все еще остается достаточно неопределенной. Подробнее узнать, что пишет по этому поводу Ричард Столлмен, основоположник Free Software Foundation, можно по адресу: *http://is.ifmo.ru/foundation/gnu.htm*. – *Примеч. науч. ред.*

узер Konqueror может отображать содержимое документов PDF с помощью программы просмотра документов этого типа – KPDF, благодаря чему отпала необходимость встраивать компоненты просмотра документов PDF в сам броузер. То же самое можно сказать о пакете офисных приложений KOffice (*http://koffice.kde.org*), который будет рассматриваться в главе 8. Так, например, текстовый процессор KWord может отображать встроенные в документ таблицы, созданные в приложении электронных таблиц KSpread.

KDE постоянно развивается, и каждые несколько месяцев команда разработчиков KDE выпускает так называемый официальный релиз, который считается устойчивым и пригодным для конечных пользователей. Команда KDE выпускает эти релизы в виде исходных текстов, а большинство распространителей Linux примерно за день-два собирают из них двоичные пакеты, более простые в установке. Если же вам не терпится поработать с самой свежей версией KDE и вас не пугают эпизодически обнаруживающиеся ошибки, можно скачать ежедневно обновляющийся вариант KDE, но это развлечение не для слабонервных. Когда писалась эта книга, текущей стабильной версией была 3.4.2.[1] Чтобы оставаться постоянно в курсе развития событий, касающихся KDE, посещайте почаще официальный сайт проекта KDE: *http://www.kde.org*.[2]

Общие характеристики

Одна из задач, стоявших перед командой разработчиков KDE, заключалась в реализации возможности настройки всего, что может быть настроено в KDE, через графические окна диалогов. В основе системы настроек лежат обычные текстовые файлы, в которых настройки представлены в чрезвычайно простом формате `parameter=value`, поэтому при желании вы можете, как и прежде, вручную вносить изменения в файлы конфигурации. Даже самые опытные пользователи обычно соглашаются с тем, что для выполнения такого простого действия, как изменение цвета фона, быстрее все-таки нажать несколько кнопок, чем искать в руководстве синтаксис команды для задания цвета фона, открывать файл конфигурации, редактировать его и перезапускать менеджер окон.

Помимо легкости в конфигурировании KDE предоставляет некоторые другие возможности, о которых ранее в Linux никто не слышал. Например, система полностью интегрирует в рабочий стол доступ в Интернет. При этом менеджер файлов выступает в роли веб-броузера, и просмотр файлов на каком-либо FTP-сайте почти не отличается от навигации по файлам на локальном диске. Можно перетаскивать на рабочий стол значки, представляющие адрес интернет-ресурса, и обращаться к ним позже. KDE интегрирует с рабочим столом поисковые машины и другие ресурсы Интернета и даже позволяет с легкостью определить свои предпочтительные поисковые машины и ссылки в Интернете. Кроме того, почти все приложения KDE способны открывать и сохранять файлы на удаленных системах, осуществляя доступ к ним не только с помощью протоколов FTP или HTTP, но и с использованием шифрования SSH, а также получать и записывать файлы изображений на цифровые фотокамеры.

---

[1] На момент подготовки этого текста – версия 3.5.5. – *Примеч. науч. ред.*

[2] Русскоязычный портал KDE: *http://www.kde.ru/*. – *Примеч. науч. ред.*

Функция перетаскивания (drag-and-drop), обычная в Windows или Macintosh, также широко применяется в KDE. Например, чтобы открыть файл в текстовом редакторе, можно найти его значок в окне менеджера файлов и перетащить в окно редактора. При этом не важно, где находится файл. Если он расположен на удаленном сервере, KDE автоматически загрузит файл, прежде чем открыть его в текстовом редакторе или в любом другом приложении, с помощью которого этот файл должен быть открыт. То же относится к мультимедийным файлам. Щелкнув на значке MP3-файла на удаленном сервере, можно загрузить его в фоновом режиме и воспроизвести на локальной машине.

Хотя страницы справочного руководства очень хорошо подходят для быстрого доступа к краткой информации о системных библиотеках, которыми обычно пользуются программисты, они не очень удобны для документации, предназначенной конечному пользователю. Поэтому KDE использует стандартные HTML-файлы (сгенерированные «на лету» из XML-файлов) и поставляется с программой просмотра справки – справочным центром KDE (KDE Help Center). Эта программа позволяет просматривать также страницы руководства manual и файлы Info, так что вы можете обращаться ко всей документации в вашей системе из одного приложения. Кроме того, большинство приложений KDE поддерживают контекстную справку.

В последние версии X Window включена возможность, называющаяся *управление сеансом*. Когда завершается работа с графической средой X (при выключении или перезагрузке компьютера), приложения, которые поддерживают возможность управления сеансом, появляются вновь на своих местах и в той же конфигурации. К сожалению, эта очень удобная возможность редко поддерживалась X-приложениями, но зато KDE использует ее весьма широко. KDE предоставляет менеджер, управляющий сеансом, и все приложения KDE пишутся с учетом этой возможности. Кроме того, KDE поддерживает и другие современные функциональные возможности X11, поддерживаемые сервером X, такие как сглаживание шрифтов (в большинстве X-серверов это реализовано посредством так называемого *расширения отображения – RENDER extension*).

В KDE включен свой менеджер окон `kwin`, причем отличный. Но это лишь часть KDE. Помимо менеджера окон в состав KDE входят файловый менеджер, веб-броузер, панель, пейджер, центр управления настройками рабочего стола и многое другое. При желании можно даже запустить KDE с другим оконным менеджером, но при этом потеряются некоторые из возможностей интеграции. Кроме того, вместе с KDE поставляется тьма приложений, от полного комплекта офисных программ и средств просмотра PostScript и PDF до мультимедийных программ и игр.

Вы можете подумать: «Это все звучит красиво, но у меня есть пара обычных приложений X, которые я хочу запускать». В этом случае вам будет приятно услышать, что такая возможность у вас сохраняется. Все ваши приложения X можно запускать на рабочем столе KDE, при этом KDE даже предоставляет, насколько это возможно, некоторые средства для интегрирования их в общую рабочую среду. Например, по вашему желанию KDE может попытаться перенастроить ваши приложения на использование тех же цветов, которыми пользуется весь рабочий стол. Конечно, приложения, не поддерживающие KDE, не поддерживают и некоторые из таких возможностей KDE, как перетаскивание или управление сеансом, но вы можете продолжать пользоваться привычными программами,

пока не появятся какие-нибудь приложения KDE, выполняющие такие же функции (или, возможно, KDE-версии ваших любимых программ).

Установка KDE

Большинство современных дистрибутивов Linux поставляются вместе с KDE, но если у вас не такой дистрибутив или вам нужна более новая версия KDE, можно скачать ее из Интернета. Все связанное с KDE, включая документацию, снимки экранов и адреса для загрузки программ, вы найдете по адресу *http://www.kde.org*. Поскольку FTP-сайт проекта KDE *ftp://ftp.kde.org* часто перегружен, может быть удобнее попробовать обратиться к зеркалам. Список зеркал можно найти на странице *http://www.kde.org/mirrors/*.

В состав KDE входят следующие пакеты:

aRts

aRts (от «real-time sequencer» – секвенсор реального времени) составляет основу большинства мультимедийных приложений KDE.

kdelibs

Библиотеки KDE. Сюда входит основной каркас приложения, ряд графических элементов графического интерфейса, система конфигурации, система отображения HTML и многое другое. Без этого пакета KDE работать не будет.

kdebase

В этом пакете находятся основные приложения KDE, которые превращают обычный рабочий стол в рабочий стол KDE, включая файловый менеджер/веб-броузер, менеджер окон и панель. Этот пакет определенно необходим для использования KDE.

kdegames

Пакет содержит игры, включая карточные, стратегические и прочие. Этот пакет, возможно, захотят установить многие (исключительно для знакомства с системой, конечно).

kdegraphics

Программы, связанные с графикой, такие как программы просмотра файлов *dvi*, PostScript и редактор пиктограмм.

kdeutils

Этот пакет содержит некоторые рабочие средства, такие как текстовые редакторы, калькулятор, менеджер печати и тому подобные.

kdemultimedia

Как следует из названия пакета, он содержит программы для работы с мультимедиа, такие как CD-плеер, MIDI-плеер, MP3-плеер и даже Karaoke-плеер.

kdenetwork

Здесь вы найдете программы для работы в Интернете, включая программу чтения телеконференций и некоторые средства управления сетью. Программа чтения электронной почты не входит в состав этого пакета, она была перемещена в пакет kdepim.

kdeadmin

Этот пакет содержит некоторые программы для системного администратора, включая программы управления пользователями, редактор уровней загрузки и программу для резервного копирования.

kdepim

Этот пакет в настоящее время многими рассматривается как главная часть KDE. *kdepim* содержит приложения для управления персональной информацией, наиболее примечательным из которых является Kontact, интегрирующий в себе такие приложения, как полнофункциональный менеджер персональной информации KOorganizer, пакет для работы с электронной почтой KMail, адресная книга, программное обеспечение для синхронизации с PDA и многие другие удобные инструментальные средства.

kdeedu

Как подсказывает название, этот пакет содержит ряд образовательных программ – от средств, помогающих заучивать слова, до программ, рассказывающих о движении звезд и планет.

kaccessibility

Этот пакет содержит инструменты, которые облегчают работу с компьютером для людей с ограниченными возможностями, как, например, экранная лупа. Цель проекта KDE состоит в достижении полного соответствия американскому закону о физических недостатках (Americans with Disabilities Act).

kdeartwork

В этом пакете собраны компоненты для KDE – результат художественного творчества. Сюда входят различные наборы ярлыков, обоев рабочего стола и тому подобное.

kde-i18n

Большой набор пакетов, названия которых начинаются с *kde-i18n-*. Каждый из них содержит перевод всех текстовых элементов, имеющихся в KDE, на один из конкретных языков. В мире KDE в качестве языка по умолчанию используется английский (американский). Таким образом, если вам нужен только этот язык, нет нужды устанавливать любые другие пакеты *kde-i18n-*. Однако, если вы живете в Великобритании и правописание американского английского режет вам глаз, можете установить пакет интернационализации для британского английского.

kdetoys

В этом пакете находится множество небольших программ, которые не несут практической пользы и служат лишь для забавы. Попробуйте, к примеру, запустить программу AMOR – Amusing Misuse of Resources (Напрасное расходование ресурсов).

kdewebdev

Этот пакет наверняка захотят установить разработчики, занимающиеся созданием веб-страниц. В нем они найдут замечательные инструменты, как, например, HTML-редактор Quanta.

koffice

Это законченный пакет полноценных офисных программ. Кое-где в нем встречаются шероховатости, но многие уже используют его в повседневной деятельности.

Цикл выпуска новых версий KOffice не зависит от цикла выпуска новых версий KDE. На момент написания книги текущая версия имела номер 1.3.5.[1] Вся информация о KOffice есть по адресу *http://koffice.kde.org*.

Инструментальные средства разработки

В состав KDE входит целый набор пакетов для тех, кто занимается разработкой приложений. *kdesdk* содержит инструментальные средства, сценарии и информацию для разработчиков KDE-программ (если планируется разработка собственных приложений для KDE, желательно было бы посетить сайт *http://developer.kde.org*), *kdebindings* содержит набор библиотек, позволяющих использовать возможности KDE из других языков программирования, отличных от C++, и, наконец, *KDevelop* – полноценная интегрированная среда разработки, причем для разработки не только KDE-приложений, но и программ любого типа.

Кроме перечисленных выше пакетов, которые официально поставляются командой разработчиков KDE, есть сотни других программ для KDE. Текущий список доступных приложений находится по адресу: *http://www.kde.org/applications.html*.

Когда пакеты выбраны, их надо установить. Способ установки зависит от вашего дистрибутива Linux и от того, какой вариант вы предпочитаете: установку скомпилированного пакета или компиляцию из исходных кодов. Если ваш дистрибутив содержит KDE, можно также выбрать установку KDE при установке всей системы.

Когда программа уже загружена на жесткий диск, осталось выполнить всего несколько шагов. Сначала следует убедиться, что каталог, где содержатся приложения KDE, включен в переменную окружения `PATH`. По умолчанию исполняемые программы KDE находятся в */opt/kde3/bin*, но, если вы установили KDE в другой каталог[2], необходимо указать путь к этому каталогу, добавив его в переменную окружения `PATH` следующей командой:

```
export PATH=/opt/kde3/bin:$PATH
```

Для того чтобы эта установка действовала постоянно, надо добавить эту строку в файл конфигурации *.bashrc* в вашем домашнем каталоге или в системный файл конфигурации */etc/profile*.

Затем то же самое нужно проделать в отношении каталога с библиотеками KDE (по умолчанию */opt/kde3/lib*) и переменной окружения `LD_LIBRARY_PATH`:

```
export LD_LIBRARY_PATH=/opt/kde3/lib:$LD_LIBRARY_PATH
```

---

[1] Версия 1.5.2 – ко времени подготовки русского издания. – *Примеч. науч. ред.*

[2] Во многих дистрибутивах KDE устанавливается в другой каталог, например в */usr/bin*.

Почти все сделано. Осталось указать X, что вы хотите запускать рабочий стол KDE при запуске X Window. Это делается в файле *.xinitrc* в вашем домашнем каталоге. Но прежде сделайте резервную копию файла, а затем удалите все его содержимое и введите единственную строку:

```
exec startkde
```

startkde – это сценарий на языке командной оболочки, поставляемый с KDE, который просто запускает менеджер окон *kwin* и ряд системных служб. Обычно дистрибутивы устанавливают более сложные файлы *.xinitrc*, из которых могут даже запускаться приложения и службы, не входящие в KDE.

Если по каким-либо причинам планируется выполнить установку KDE более чем в один каталог, необходимо создать и настроить переменную окружения `KDEDIRS`, указав пути ко всем каталогам. Как правило, такая установка выполняется достаточно редко.

Использование KDE

Пользоваться KDE весьма просто. Большинство вещей интуитивно понятны, поэтому часто можно просто догадаться, что надо сделать. Однако мы дадим вам несколько советов по использованию KDE, чтобы воодушевить вас на дальнейшее исследование рабочего стола KDE.

Панель KDE и K-меню

При первом запуске среды KDE она выглядит, как на рис. 3.1. Вдоль нижнего края экрана видна так называемая *панель*. Панель служит нескольким целям, включая быстрый доступ к установленным приложениям и открытым окнам. При первом запуске KDE также откроет программу настройки, которая позволит изменить начальные установки.

KDE предоставляет несколько рабочих экранов (workspaces), к которым можно переходить с помощью кнопок в середине панели; по умолчанию они помечены номерами от одного до восьми. Попробуйте нажать эти кнопки. Вы увидите, что открытые вами окна видны только на первом рабочем экране, тогда как панель внизу экрана и панель задач присутствуют на экране постоянно. Теперь попробуйте, находясь в рабочем экране 2, запустить терминал, щелкнув на его значке в панели быстрого запуска. После этого перейдите в другой рабочий экран. Вы обнаружите, что терминал виден, только когда вы находитесь в рабочем экране 2, но его значок виден на панели задач во всех рабочих экранах. Если, находясь в любом другом рабочем экране, щелкнуть на значке терминала в панели задач, произойдет немедленный переход к экрану, где был запущен терминал.

Другая удобная функция – это маленький значок с изображением кнопки в левом верхнем углу окна терминала. Нажмите его и перейдите в другой экран. Терминал виден теперь в каждом рабочем экране – вы прикрепили его к фону рабочего стола.

Если такое прикрепленное окно вам надоело, нажмите кнопку снова и отцепите окно, а если вы хотите совсем его закрыть, нажмите кнопку в правом верхнем углу с маленьким крестиком.

Рис. 3.1. *Внешний вид рабочего стола KDE*

KDE обладает огромным числом настраиваемых параметров, и оформление окон – лишь один из них. Если вы не видите кнопки прикрепления окна ко всем рабочим экранам, это означает, что при текущих настройках оформления окон KDE в вашем дистрибутиве кнопка просто не видна. В этом случае можно щелкнуть левой кнопкой мыши на значке в левом верхнем углу окна и выбрать пункт меню To Desktop (На рабочий стол)→All Desktops (Все рабочие столы).

Есть еще много вещей, которые можно проделывать с окнами в KDE, но сейчас мы перейдем к краткому введению в так называемое *K-меню*. Оно открывается щелчком на значке в левом углу панели, где нарисована буква K с шестеренкой. Помимо некоторых параметров для настройки самого K-меню и панели, вы найдете там все установленные приложения KDE, сгруппированные в виде подменю. Чтобы запустить одно из таких приложений, просто выберите элемент меню.

Мы обещали, что вы сможете запускать ваши старые приложения X на вашем рабочем столе KDE. Вы можете запустить их, открыв окно терминала и введя название приложения в командной строке либо нажав комбинацию клавиш Alt+F2 и введя название приложения в маленькой командной строке, которая появится в центре экрана. Однако, немного поработав, вы сможете включить приложения, не являющиеся приложениями KDE, в K-меню и панель быстрого запуска, в результате чего они будут представлены значками, щелчком на которых вы сможете запустить связанные с ними программы.

В зависимости от того, как вы установили KDE, очень может быть, что среди элементов меню уже есть подменю, содержащее приложения, не поддерживаю-

щие KDE. Если этого подменю нет, запустите приложение KAppfinder, находящееся в подменю System (Система), или запустите из командной строки утилиту `kappfinder`. Эта программа отыщет в вашей системе приложения, которые входят в ее базу данных, и интегрирует их в рабочий стол KDE, создав для каждого приложения файл *.desktop*. Если нужной программы нет в базе данных KAppfinder, вам придется самостоятельно написать такой файл *.desktop*, однако для этого, как, впрочем, и всегда в KDE, вам нужно будет лишь ввести требуемую информацию в нескольких диалогах. См. документацию KDE по адресу *http://www.kde.org/documentation/index.html*.

По умолчанию панель быстрого запуска уже содержит значки для запуска наиболее часто используемых программ, но можно легко добавить собственные. Для этого достаточно щелкнуть правой кнопкой мыши где-нибудь на свободном месте панели и выбрать пункт меню Add to Panel (Добавить на панель)→Application (Приложение). Появится копия K-меню. Найдите приложение, значок которого вы хотите добавить на панель, и выделите его так же, как если бы вы хотели его запустить. KDE добавит на панель значок этого приложения. Можно даже добавлять на панель целые подменю, выбирая первый пункт в подменю Add this Menu (Добавить это меню). В этом случае к значку добавляется небольшая черная стрелка, указывающая на то, что щелчок на значке откроет меню, а не запустит приложение.

Кроме кнопок запуска приложений на панель можно добавить еще много чего, например *апплеты* панели – маленькие программы, которые созданы специально для работы в составе панели и не могут быть запущены как самостоятельные приложения. В результате исследования контекстного меню панели можно найти немало интересного.

Пространство на панели небесконечно, поэтому может возникнуть необходимость удалить некоторые значки программ, которыми вы не часто пользуетесь. Щелкните правой кнопкой мыши на значке и выберите пункт контекстного меню Remove (Удалить кнопку) (пункт Remove (Удалить кнопку) будет содержать имя программы, кнопка вызова которой удаляется). При этом будет удалена не сама программа, а только ее значок. Вообще, в KDE щелчок правой кнопкой мыши обеспечивает доступ к множеству функций!

Центр управления KDE

Теперь мы покажем, как настроить рабочий стол KDE по своему вкусу. Как мы обещали, для этого не потребуется редактировать никакие файлы конфигурации.

Настройка производится через центр управления KDE (KDE Control Center), который запускается из K-меню. (В некоторых дистрибутивах пункт запуска центра управления можно найти на верхнем уровне иерархии K-меню, в других, таких как Debian, в одном из подменю, например Settings (Настройка). Все параметры настройки сгруппированы по типам операций. Начав работу с центром управления, вы увидите группы верхнего уровня. Нажав значок +, можно открыть группу и обратиться к ее элементам.

Настройка фона. Для примера мы поменяем цвет фона. Для этого надо открыть группу Appearance & Themes (Внешний вид и темы) и выбрать пункт Background (Фон). Появится окно для настройки фона (рис. 3.2).

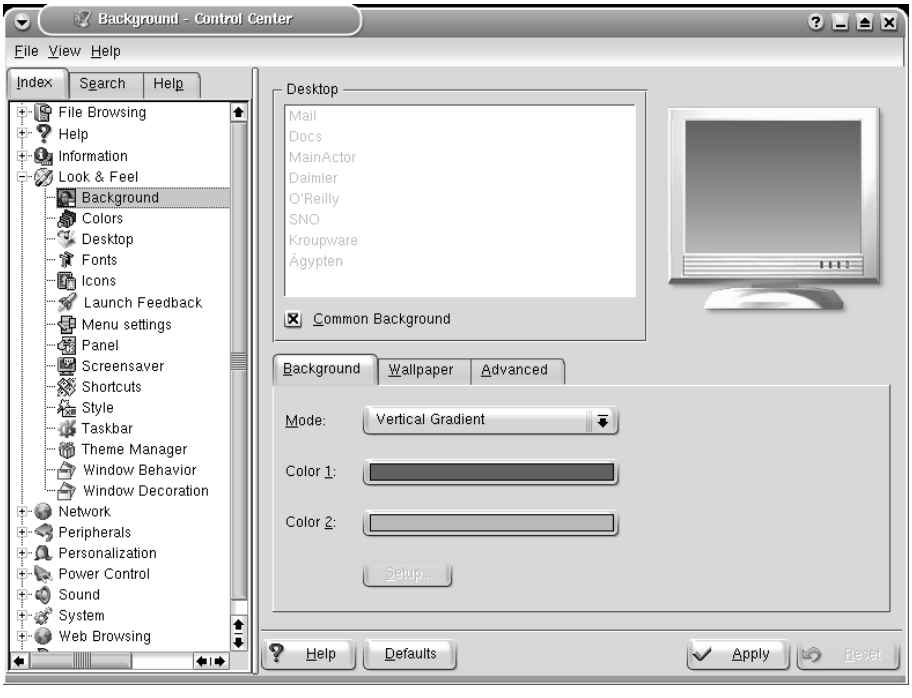


Рис. 3.2. *Настройка фона рабочего стола KDE*

Можно выбрать одноцветный фон, двухцветный фон, одну из разновидностей градиентной окраски, когда один цвет плавно переходит в другой, обои (готовые или из ваших графических изображений) и эффект слияния, в котором объединены несколько вариантов. Чтобы выбрать цвет, нужно щелкнуть на одной из двух кнопок, в результате появится диалог, где можно выбрать цвет по своему вкусу. После закрытия диалога новый цвет можно увидеть в области предварительного просмотра, в верхнем правом углу окна настройки. При настройке KDE вы часто будете встречать возможность предварительного просмотра вносимых изменений. Однако можно посмотреть выбранные настройки и на полном экране, для чего нужно щелкнуть на кнопке Apply (Применить) в нижней части диалога, и изменения вступят в силу. При этом нет необходимости перезапускать рабочий стол. Если после настройки никаких изменений на рабочем столе не произошло, посмотрите, снят ли флажок Picture (Изображение). Когда этот флажок установлен, фон рабочего стола может оказаться просто невидим за лежащим сверху изображением. Попробуйте выбрать какое-либо изображение и поэкспериментировать с эффектами слияния (например, с градиентом), чтобы добиться желаемого эффекта комбинирования изображения и цвета фона.

Если вы предпочитаете одноцветный экран, установите флажок No Picture (Нет изображения), а в раскрывающемся списке Mode (Цвета) выберите значение Single Color (Один цвет). Вы увидите, что вторая кнопка выбора цвета стала неактивной. Выберите нужный цвет первой кнопкой.

Если среди ваших рисунков или поставляемых в составе дистрибутива нет такого, который хотелось бы видеть в качестве фонового изображения на своем рабочем столе, можно щелкнуть на кнопке Get New Wallpapers (Получить новые обои) и получить доступ к огромному количеству обоев, созданных пользователями KDE, из которых можно подобрать фоновый рисунок на любой вкус.

Настраивать фон можно и дальше, но сейчас мы займемся кое-чем другим, а точнее – настройкой стилей и цвета окон.

Настройка стилей и цвета окон. В обычных менеджерах окон можно настроить только цветовое оформление окна, но не его содержимое. KDE – это не обычный менеджер окон, а интегрированный рабочий стол, поэтому можно изменить не только цвет и другие параметры оформления окна, которые контролируются оконным менеджером, но и параметры отображения содержимого окна, управляемые приложениями. Сейчас мы немного поработаем над их внешним видом.

В центре управления откройте группу Look & Feel (Внешний вид и темы) и выберите пункт Colors (Цвета). Вы увидите окно предварительного просмотра и список, в котором можно выбрать цветовую схему. KDE работает не через настройку отдельных цветов, а через определение так называемых *цветовых схем*. Это делается потому, что изменять только один цвет бессмысленно: все цвета должны сочетаться друг с другом, чтобы на них можно было смотреть без содрогания.

KDE позволяет создавать собственные цветовые схемы, но это требует некоторых знаний из области психологии человеческого зрения. Поэтому мы предлагаем выбрать одну из имеющихся цветовых схем. После того как вы оценили схему в окне предварительного просмотра, нажмите кнопку Apply (Применить) и посмотрите, как все запущенные приложения изменят цвет – без всякой перезагрузки. Для пользователей Windows в этом нет ничего необычного, а в UNIX до появления KDE ничего подобного не было.

Такие же функции применяются и в отношении других настроек. Например, откройте группу Look & Feel (Внешний вид и темы) и выберите пункт Style (Стиль). Здесь вам будет представлен большой выбор стилей оформления рабочего стола. Стиль определяет способ отображения элементов интерфейса пользователя, например, как в Windows (стиль MS Windows 9x), как в Motif (стиль Motif), как на рабочих станциях SGI (стиль SGI) или что-нибудь оригинальное, как, например, стили Light (Освещение) и фавориты KDE – стили Plastik (Пластик) и Keramik (Керамика).[1] Сделав свой выбор, можно нажать кнопку Apply (Применить) и посмотреть, как все запущенные приложения изменят свой стиль. Кстати, то же самое относится и к шрифтам, которые можно выбрать в группе Fonts (Шрифты).

Интернационализация. Мы не можем описать все настраиваемые элементы KDE, ибо тогда в книге не осталось бы места для остальных тем. Но еще одну из настроек мы не можем обойти вниманием. Она вам особенно понравится, если английский язык не является вашим родным языком или вы часто пользуетесь другими языками.

Перейдите на страницу Country & Language (Страна/область и язык) в группе Regional & Accessibility (Региональные и специальные возможности) (рис. 3.3). Здесь

[1] Если вам до сих пор кажутся странными названия некоторых терминов KDE, вспомните первую букву в названии рабочего стола.

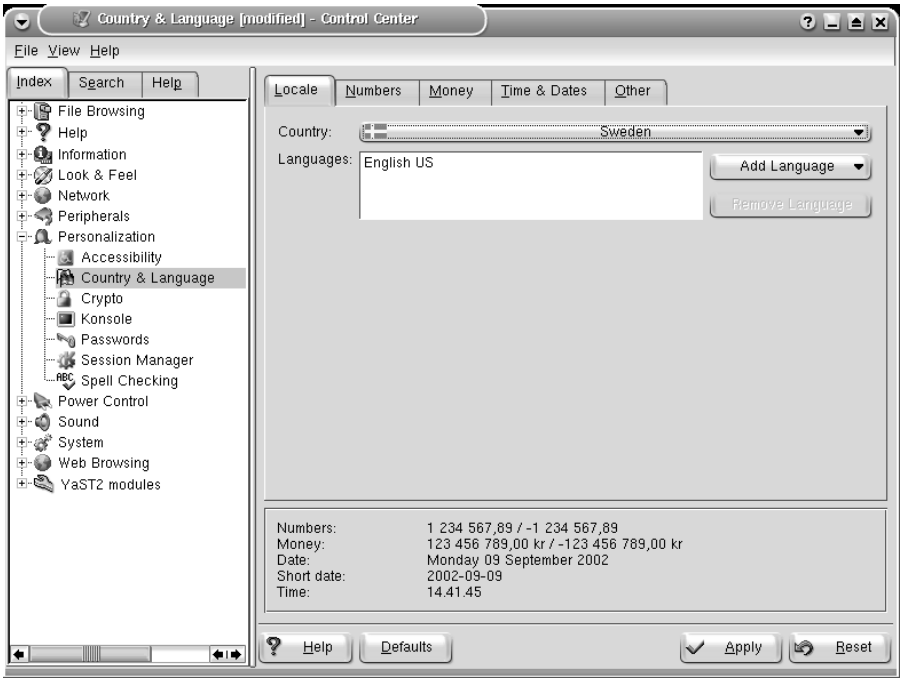


Рис. 3.3. *Настройка языка рабочего стола KDE*

можно выбрать страну и язык, которые будут использоваться рабочим столом и приложениями KDE. На текущий момент KDE позволяет выбирать более чем из 80 национальных настроек и языков. Учтите, что для выбора конкретного языка должен быть установлен пакет интернационализации для этого языка. Эти пакеты можно загрузить с FTP-сервера KDE (как описано выше) или установить с носителя дистрибутива.

Вы можете удивиться, зачем может понадобиться выбирать более одного языка. Причина в том, что программы KDE переводятся добровольцами. При этом не все приложения переводятся одновременно, а некоторые и вовсе не переведены. Поэтому какое-либо приложение может оказаться недоступным на том языке, который выбран в качестве основного (верхнего в списке Languages (Языки)). В этом случае для него автоматически используется второй из выбранных языков, а в случае неудачи будет выбран третий и т. д. Если ни с одним из этих трех языков приложение работать не может, будет выбран английский (США), который всегда присутствует в системе.

Говоря о различных языках, имеет смысл упомянуть о возможности настройки раскладки клавиатуры. Большинство европейских языков, даже основанные на латинском алфавите, имеют специальные символы, которые недоступны на других клавиатурах или которые не так просто ввести. В состав KDE входит небольшая программа, которая позволит быстро изменить раскладку клавиатуры. Разумеется, программа не может изменить маркировку на клавишах, но быстрое изменение раскладки само по себе может оказаться очень удобным для тех, кто по-

стоянно путешествует по миру, как некоторые из авторов этой книги. Чтобы включать эту функциональную возможность, нужно перейти на страницу Keyboard Layout (Раскладка клавиатуры) в группе Regional & Accessibility (Региональные и специальные возможности) и установить флажок Enable keyboard layouts (Включить переключение раскладок клавиатуры). Затем из имеющихся раскладок необходимо выбрать активные раскладки, которые предполагается использовать. После щелчка на кнопке Apply (Применить) в правой части панели (которая называется *системным лотком* (*system tray*)) появится кнопка с изображением флага. Щелчком мыши на этом флаге можно будет быстро менять раскладку клавиатуры.

Можно еще долго говорить об удобствах рабочего стола KDE, но лучше начать самостоятельное изучение. Кроме очевидных и интуитивных функций есть и менее очевидные, но тем не менее очень полезные возможности, поэтому не стесняйтесь и почитайте документацию по адресу *http://www.kde.org/documentation/index.html*.

Приложения KDE

Для KDE существуют тысячи программ – от базовых утилит, таких как *konsole* (эмулятор терминала) и *OClock* (часы), до редакторов, средств разработки приложений, игр и мультимедийных приложений. В нашей книге мы можем охватить лишь малую часть из того множества программного обеспечения, которое доступно для KDE. В этом разделе мы расскажем о тех приложениях, пользоваться которыми должны уметь все пользователи KDE. Это не обязательно самые замечательные и впечатляющие программы, но их определенно следует иметь в своем арсенале рабочих приложений.

Кроме того, не забывайте, что приложений KDE значительно больше, чем мы здесь сможем перечислить. Некоторые из них, такие как KWord (текстовый процессор) или Kontact (менеджер персональной информации и электронной почты), встретятся вам в других местах книги. Однако многим другим приложениям места в книге просто не нашлось. Поэтому покопайтесь в архивах Linux, и вы найдете сотни замечательных KDE-программ.

Кроме того, следует постоянно помнить, что если для решаемой вами задачи нет KDE-программы, можно воспользоваться классическим X-приложением, если таковое имеется. Оно не будет выглядеть так изящно и так хорошо интегрироваться в общее окружение, но все равно будет функционировать на рабочем столе KDE.

konsole: ваше главное окно

Мы начнем наше исследование приложений X с рабочей лошадки – в терминале вам, возможно, придется проводить большую часть времени. Это просто окно, в котором содержится командная оболочка UNIX. Оно выводит приглашение к вводу, принимает команды и прокручивается подобно терминалу.

Традиционно классическим эмулятором терминала UNIX являлось приложение *xterm*. В окружении рабочего стола KDE его заменила программа *konsole*.

Возможно, некоторые из вас усмехнутся, глядя на то, как, купив дорогой цветной монитор и установив мегабайты графического программного обеспечения, мы сидим перед эмулятором старого терминала VT100. Но не надо забывать, что это Linux, а не просто операционная система типа «укажи и щелкни». Здесь есть и замечательные графические приложения, но вам наверняка придется уделять много времени работе с текстами, а интерфейс командной строки – это самое мощное средство для таких задач. Более детально об этих задачах мы поговорим в главе 4.

Итак, посмотрим на окно *konsole* (рис. 3.4), содержащее несколько команд.

Рис. 3.4. *Окно konsole*

Запуск konsole

Запустить *konsole* можно несколькими способами, как и все программы KDE:

- Запустить из панели, если там есть значок *konsole*. В большинстве дистрибутивов он устанавливается по умолчанию.
- Вызвать через K-меню, где вызов *konsole* производится выбором пункта Utilities (Утилиты)→System (Система)→Konsole (Терминал).
- Нажать комбинацию клавиш Alt+F2 и в открывшемся маленьком окне команд ввести `konsole`.
- Если программа *konsole* уже запущена, можно ввести в ней команду `konsole` и получить еще одно окно терминала или выбрать в окне программы пункт меню Session (Сеанс)→New Shell (Новое окно).

При открытии окна *konsole* появляется окно диалога Tip of the Day (Совет дня), в котором даются полезные советы, касающиеся работы с программой *konsole*. Его можно отключить, но мы советуем на пока его оставить, чтобы время от времени узнавать некоторые полезные вещи. Можно просмотреть все советы, многократно щелкая по кнопке Next (Далее) в этом диалоге. У многих приложений KDE есть такое же окно Tip of the Day (Совет дня).

В одном окне *konsole* можно запускать несколько сеансов. Новый сеанс можно запустить, просто выбрав его тип в меню Session (Сеанс) или щелкнув на кнопке New (Новый), находящейся на панели вкладок. После этого можно переключаться между сеансами с помощью панели вкладок или меню View (Вид). Если панель

вкладок отсутствует в окне программы, выберите в меню пункт Settings (Настройка)→Tab Bar (Панель вкладок) и далее пункт Top (Вверху) или Bottom (Внизу).

Операции удаления и вставки

На самом деле *konsole* предлагает значительно больше, чем терминал VT100. Одной из возможностей является мощная операция удаления и вставки.

Взглянем еще раз на рис. 3.4. Допустим, нам не нужен каталог *notes*; вместо этого мы хотим посмотреть на *~/perl_example/for_web_site*.

Выберем сначала интересующую нас часть команды `cd`. Поместите курсор мыши слева от символа `c` в имени команды `cd`. Нажав левую кнопку мыши, перемещайте мышь, пока не будет выделен символ слэша после `example`. Результат показан на рис. 3.5.

Когда выделенная область охватит нужное количество символов, нажмите среднюю кнопку мыши.[1] *konsole* вставит выделенный текст в следующую командную строку (рис. 3.6). Теперь можно набрать оставшуюся часть пути к каталогу *for_web_site* и нажать клавишу Enter, чтобы выполнить команду.

В окне можно выбрать все что угодно: как ввод, так и результаты выполнения команд. Для выбора целых слов вместо символов нужно выполнить двойной щелчок левой кнопкой мыши. Целая строка выбирается тройным щелчком. Вы можете выбрать также и несколько строк. Это не требуется при вводе команд, но удобно, если используется редактор *vi* и надо часто переносить фрагменты текста между окнами.

Обратите внимание, если вам привычнее копировать текст путем перетаскивания (drag-and-drop), *konsole* поддерживает и такой стиль.

Операции копирования и вставки текста возможны даже между *konsole* и другими приложениями KDE с графическим интерфейсом. Например, при просмотре

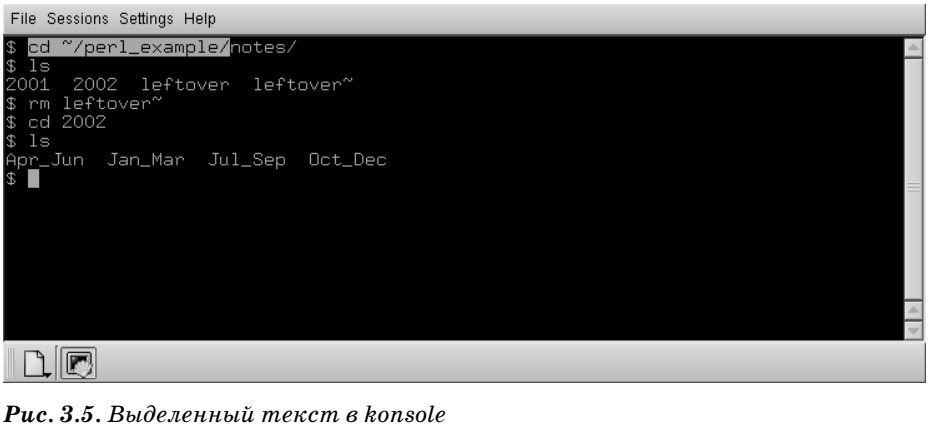


Рис. 3.5. *Выделенный текст в konsole*

[1] Если у вас двухкнопочная мышь или щелчок средней кнопкой не производит никакого эффекта, прочитайте раздел «Настройка X.org» в главе 16, где описано, как выполнить настройку мыши.

Рис. 3.6. *Окно konsole после вставки текста*

содержимого каталога с помощью файлового менеджера или веб-броузера Konqueror можно перетаскивать значки каталогов в окно *konsole*, а в ответ на это будет предложено просто вставить имя каталога или файла или предварить его одной из четырех команд: `cd`, `cp`, `mv` или `ln`.

Дополнительные приемы работы с konsole

Очень многие характеристики *konsole* можно настраивать. Можно выбирать шрифты, цветовые схемы, показывать полосу прокрутки слева, справа или вообще убрать ее и т. д. К наиболее часто используемым настройкам можно обращаться через меню Settings (Настройка), а если вы не можете найти требуемое, выберите пункт меню Settings (Настройка)→Configure Konsole (Настроить konsole). В результате будет открыт диалог, где можно установить интервал между строками, изменить частоту мерцания курсора и т. д.

Очень полезна такая функция *konsole*, как возможность сторожить вывод или его отсутствие в одном из сеансов.

Для чего это нужно? Представьте себе, что вы разрабатываете большую программу, которая долго компилируется. Непрограммисты могут представить себе загрузку большого файла в окне терминала с помощью *wget* или расчет сложного образа POVRAY. Пока выполняется компиляция, хотелось бы заняться чем-то другим (в конце концов, для чего существуют многозадачные операционные системы?), скажем, начать составление письма в почтовом клиенте KDE. Обычно вы должны время от времени переключаться в окно консоли и проверять, не закончилась ли компиляция, а затем продолжать разработку программы. С помощью сторожа можно получать зрительное или звуковое уведомление по завершении компиляции. Просто переключитесь в сеанс, за которым вы хотите следить, и выберите пункт меню View (Вид)→Monitor for Silence (Монитор простоя). Вы получите уведомление, как только компилятор в течение некоторого времени не будет выдавать сообщения, и это переключит ваше внимание от почтового клиента к окну *konsole*. Конечно, можно следить и за появлением вывода, а не за его прекращением, например при длительной сетевой операции, которая не выводит индикатор хода процесса.

Часы

Разве может ваш экран быть полноценным, если его не украшают небольшие часы, показывающие, сколько времени вы убили, настраивая внешний вид вашего экрана? Часы могут быть самыми разными: квадратными или круглыми, со стрелками или электронными, маленькими или большими. Вы даже можете заставить их бить в куранты.

В KDE есть часы нескольких типов, но обычно запускают небольшое приложение на панели, поскольку независимо от разрешения место на экране всегда в цене. По умолчанию часы появляются в правом нижнем углу экрана в пределах панели (это так называемый *апплет панели*, то есть маленькое приложение, исполняющееся внутри панели). Если дистрибутив создал другие настройки, можно щелкнуть правой кнопкой в любом месте фона панели, выбрать в контекстном меню пункт Add Applet to Panel (Добавить апплет на панель) и в открывшемся диалоге выбрать пункт Clock (Часы), в результате чего часы появятся на панели. Если вы хотите поместить часы в другом месте панели, щелкните правой кнопкой на маленькой полосатой ручке слева от часов, выберите в появившемся контекстном меню пункт Move Clock (Переместить часы) и переместите часы мышью в нужное местоположение. Остальные объекты панели автоматически освободят место для часов.

У апплета часов есть несколько режимов, которые устанавливаются путем щелчка правой кнопкой на самих часах и выбора в контекстном меню пункта Type (Тип), а затем нужного режима. Есть простые, цифровые, аналоговые и, что примечательно, неточные (fuzzy) часы. Неточные часы – для тех, кто не любит, чтобы его подгоняло время. Например, если запустить «неточные» часы, они могут показать время Middle of the week (Середина недели). Если это покажется слишком неточным, можно выбрать в контекстном меню часов пункт Configure Clock (Настроить часы) и в открывшемся диалоге на вкладке Appearance (Внешний вид) установить степень неточности. Например, допустим, текущее время 9:53, в четверг, тогда четыре степени неточности дадут «без пяти десять», «десять ровно», «позднее утро» и уже упомянутую «середину недели».

Кроме того, приложение часов позволяет настроить формат отображения даты и времени, а также установить системное время (для этого потребуются права *root*; если вы зарегистрированы как обычный пользователь, появится окно с просьбой ввести пароль *root*). Можно даже скопировать текущие дату и время в системный буфер обмена в самых разных форматах.

KGhostview: просмотр файлов PostScript и PDF

Стандарт Adobe PostScript стал одним из самых популярных форматов для обмена документами в компьютерном мире. В этом формате распространяется множество научных работ. Руководства из Linux Documentation Project тоже распространяются в виде PostScript среди прочих форматов. Этот формат особенно удобен для тех, у кого нет времени на форматирование или имеется достаточная полоса пропускания сети для передачи огромных файлов. Создавая собственные документы с помощью *groff* или TEX, вы, наверное, захотите увидеть их на экране, прежде чем портить дорогостоящую бумагу, распечатывая их.

KGhostview, приложение KDE, предлагает удобную среду для просмотра файлов PostScript в X Window System, в которой помимо файлов PostScript можно так-

же просматривать файлы в формате Adobe PDF. Однако специально для просмотра документов формата PDF в KDE существует другое приложение – *kpdf*. KGhostview в действительности является лишь удобным интерфейсом к более старому приложению Ghostview, поэтому описываемые здесь функции можно выполнять и с помощью Ghostview. Но пользователь значительно комфортнее чувствует себя в KGhostview, поэтому мы здесь и опишем эту программу.

Программа KGhostview очень проста; она вызывается с именем файла, который надо просмотреть, например:

```
eggplant$ kghostview article.ps
```

Можно также просто щелкнуть на значке любого файла PostScript или PDF в каком-либо месте KDE.

Поскольку сейчас мы интересуемся только просмотром имеющихся файлов, нас не должны беспокоить преимущества самих PostScript и PDF. Оба могут считаться стандартами для многих программ, способных записывать файлы в этих форматах (а некоторые могут и читать); оба созданы одной компанией, Adobe Systems. PDF несколько более переносим между платформами и самодостаточен, так как может даже содержать шрифты, необходимые для показа документа. Кроме того, PDF лучше известен в Microsoft Windows и Macintosh, поэтому в Интернете у вас больше шансов встретить файлы PDF, а не файлы PostScript. И, наконец, PostScript в действительности предназначен для печати, тогда как в PDF есть некоторые функции для интерактивного просмотра, такие как значки страниц, гиперссылки и т. п.

KGhostview не является совершенным средством просмотра PDF, хотя для большинства документов его достаточно. Если возникают проблемы с конкретным документом, можно попробовать воспользоваться Adobe Acrobat Reader (не являющимся свободно распространяемым программным обеспечением, но допускающим бесплатную загрузку с *www.adobe.com*) либо KDE-программой *kpdf*, которая входит в состав того же пакета, что и KGhostview.

Окно KGhostview огромно; оно может легко занять большую часть вашего экрана. Первая страница документа при необходимости имеет полосы прокрутки. Как и в большинстве программ KDE, имеются панели меню и инструментов, а также средства прокрутки страниц и список страниц в левой части окна.

Как и большинство приложений X, KGhostview предлагает для основных функций и команды меню, и быстрые комбинации клавиш. Таким образом, чтобы перейти к следующей странице, можно выбрать в меню View (Вид) пункт Next Page (Следующая страница). Либо можно просто нажать клавишу PgDn (или клавишу пробела при отсутствии PgDn, как на лэптопе).[1]

Вернуться к предыдущей странице можно, выбрав пункт Previous Page (Предыдущая страница) в меню View (Вид). Для перехода к произвольной странице надо

[1] Есть тонкое различие между клавишами пробела и PgDn: клавиша PgDn всегда переносит вас на следующую страницу, тогда как клавиша пробела сначала переносит в нижнюю часть текущей страницы, если размеров окна недостаточно для отображения на экране целой страницы. Второе нажатие пробела переводит на следующую страницу.

левой кнопкой мыши нажать соответствующую цифру в колонке Page Number (Номер страницы). Для выхода выберите пункт Quit (Выход) в меню File (Файл) или просто нажмите Ctrl+Q.

В разных странах используются разные размеры страниц. Ghostview по умолчанию использует американский стандарт *letter* (его можно закомментировать в файле PostScript, что часто делается инструментами PostScript в дистрибутивах Linux, настроенных для работы в Европе). Вы можете указать другой размер в подменю Paper Size (Размер бумаги) меню View (Вид).

Ghostview позволяет увеличивать или уменьшать размер страницы. Это очень полезная функция для детального просмотра результатов форматирования. (Но учтите, что экранные шрифты отличаются от шрифтов в принтере, и поэтому расположение символов на экране Ghostview не будет точно таким же, как на бумажной копии.) Для увеличения некоторой части страницы нажмите Ctrl+«+»; для уменьшения используйте Ctrl+«-». Можно также воспользоваться кнопками панели инструментов и пунктами Zoom In/Zoom Out (Увеличить/Уменьшить) в меню View (Вид).

Можно настроить размер окна, чтобы оно точно соответствовало ширине страницы документа, выбрав пункт Fit To Page Width (По ширине страницы) в меню View (Вид).

Для распечатки страницы выберите пункт Print (Печать) в меню File (Файл) или нажмите Ctrl+P в любом месте окна. Появится стандартное окно печати KDE, которое, помимо прочего, позволяет выбрать принтер.

Можно также распечатать только текущую страницу или диапазон страниц; достаточно задать свой выбор в диалоговом окне печати. Можно объединить это с функцией выделения страниц. Меню Edit (Правка) позволяет устанавливать и снимать отметки для отдельных страниц или их групп. Помеченные страницы показываются с маленьким красным флажком в списке страниц. Если есть помеченные страницы и выбирается функция печати, в диалоговом окне автоматически заполняется список печатаемых страниц. Конечно, его можно изменить, прежде чем отправить документ на печать.

Чтение документации с помощью Konqueror

Konqueror не только является высококачественным веб-броузером и файловым менеджером, но и выполняет функцию чтения документации. Документация KDE поставляется в формате HTML, однако Konqueror способен отображать документы и в других форматах, например Info и страницы справочного руководства manual. Например, чтобы вывести страницу справочного руководства для команды `ls`, следует открыть мини-окно командной строки, нажав Alt+F2 и введя:

```
man:ls
```

KDE поймет, что была запрошена страница справочного руководства для команды `ls`, откроет окно Konqueror и покажет страницу руководства. Результат будет отформатирован гораздо красивее, чем это сделала бы команда `man` (или ее аналог в X11 `xman`).

Аналогичное происходит со страницами Info. Например, документация для компилятора GNU C *gcc* поставляется в формате *info*. Просто введите:

```
info:gcc
```

в командной мини-строке или строке ввода URL Konqueror, и появится требуемая страница Info (если, конечно, она установлена). Если ранее вас раздражала действительно неудобная программа командной строки *info* и программы типа *xinfo* тоже не радовали, данная возможность окажется весьма кстати.

Но Konqueror этим не ограничивается, когда нужно получить информацию. Хотите воспользоваться поисковым механизмом в Интернете? Чтобы найти страницы, касающиеся Tux (талисман Linux), например, на AltaVista, просто введите в командной мини-строке или строке ввода URL Konqueror:

```
av:tux
```

и появится окно Konqueror с 3 360 000 (на момент написания этих строк) результатами поиска. Можно воспользоваться и другими поисковыми механизмами. В табл. 3.1 приводятся некоторые наиболее популярные поисковые механизмы и их префиксы.

Таблица 3.1. Популярные поисковые механизмы и их префиксы

Поисковый механизм	Префикс
AltaVista	`av:`
SourceForge	`sf:`
Excite	`ex:`
Google	`gg:`
Merriam-Webster Dictionary	`dict:`

Если ваш любимый поисковый механизм не включен в конфигурацию (что в действительности весьма маловероятно), добавьте его самостоятельно: для этого нужно открыть окно Konqueror, выбрать пункт меню Settings (Настройка)→Configure Konqueror (Настройка Konqueror), и в появившемся диалоге, в разделе Enable Web Shortcuts (Сокращения Веб), можно найти все заранее настроенные поисковые механизмы и добавить свои собственные.

Запись компакт-дисков с помощью K3b

В состав KDE входит очень удобная и пользующаяся большой популярностью программа, предназначенная для записи компакт-дисков и DVD-дисков, – K3b. Если вставить в устройство записи компакт-дисков пустой CD-R или DVD-R, KDE автоматически предложит запустить программу K3b, если этого не происходит, программу можно запустить из командной строки, дав команду `k3b`. Кроме того, в большинстве дистрибутивов эту программу можно найти в K-меню.

Обычно K3b сама распознает приводы CD и DVD, но если она этого не делает, можно выбрать пункт меню Settings (Настройка)→Configure K3b (Настроить K3b) и в появившемся диалоге выбрать страницу Devices (Устройства). На этой странице можно увидеть перечень всех обнаруженных устройств чтения-записи CD и DVD, сгруппированные по двум категориям – устройства чтения и устройства записи. Если фактически имеющееся устройство отсутствует в списке, для начала можно попробовать щелкнуть на кнопке Refresh (Обновить), если это действие не даст же-

лаемого результата, тогда можно щелкнуть на кнопке Add Device (Добавить устройство) и ввести данные об устройстве вручную. Программа K3b ожидает получения имени специального файла устройства. Во многих дистрибутивах используются символические ссылки с выразительными именами, например */dev/cdrom* или */dev/cdrecorder*, указывающие на фактические файлы устройств. Если указано корректное имя файла устройства, программа K3b, как правило, в состоянии автоматически распознать его параметры, такие как скорость чтения и записи.

Окно программы K3b делится на две половины. В верхней половине отображается дерево каталогов файловой системы. В нижней половине находятся элементы выбора выполняемых задач, таких как создание нового диска DVD с данными или копирование существующего диска. Вызов других, менее распространенных действий, таких как запись созданного ранее образа на копакт-диск, можно выполнить, выбрав пункт меню File (Файл)→New Project (Создать проект).

Рассмотрим на примере порядок записи на компакт-диск резервной копии цифровых фотографий, созданных на недавно прошедших праздниках. Для этого нужно щелкнуть на ярлыке New Data CD Project (Новый проект CD с данными). В нижней половине окна появится пустой список файлов, куда можно перетащить файлы из верхней половины окна (или из окна Konqueror). Для нашего случая достаточно просто переместить каталог с фотографиями в нижний список. После того как это будет сделано, внизу окна K3b появится зеленая полоска, которая показывает, как много места займут файлы, которые уже включены в проект, на компакт-диске. Это даст возможность узнать, можно ли еще что-то записать на диск.

После того как выбор файлов для записи будет произведен, можно щелкнуть на кнопке Burn (Записать), которая находится в левом верхнем углу нижней половины окна. В результате на экране появится диалог с настройками, где можно проверить и изменить параметры записи, но в большинстве случаев лучше оставить значения параметров, принятые по умолчанию. Обычно рекомендуется устанавливать флажок Verify written data (Проверить записанные данные) на вкладке Writing (Запись); это добавит уверенности в том, что данные были записаны на диск без ошибок (хотя это вдвое увеличит время создания компакт-диска). Вероятно, в настройках на вкладке Volume Desc (Описание тома) потребуется изменить название тома (название компакт-диска) и указать свое имя в поле Publisher (Издатель). Если предполагается возможность чтения диска не только в Linux, но и в Windows, будет не лишним установить флажки Generate Rock Ridge extensions (Формировать расширение Rock Ridge) и Generate Joliet extensions (Формировать расширение joliet) на вкладке Filesystem (Файловая система). После того как все необходимые настройки будут выполнены, можно щелкнуть на кнопке Burn (Записать), расположенной в верхнем правом углу диалога. После этого можно откинуться на спинку кресла и наблюдать за процессом записи компакт-диска.

Окружение рабочего стола GNOME

Окружение рабочего стола GNOME, как и KDE, представляет собой набор программных средств, образующих полноценный рабочий стол. Как и в случае KDE, задачей GNOME было предоставить возможность запускать любые X-приложения. И KDE, и GNOME основываются на стандартах Freedesktop. Фактически различия, имеющиеся между этими двумя рабочими столами, представляют

интерес скорее для разработчиков, чем для пользователей, которые используют приложения, нимало не беспокоясь о том, для какого рабочего стола они разрабатывались.

Первоочередная задача GNOME заключается в упрощении работы. Чтобы официально стать частью рабочего стола GNOME, приложения должны соответствовать достаточно обширным требованиям, предъявляемым к пользовательскому интерфейсу. Благодаря тому, что GNOME представляет собой замечательную платформу для разработки программ на языках C, C++, Python, Java и C#, в последнее время появилось большое количество приложений третьих фирм, которые официально не входят в состав GNOME. В некоторых случаях (особенно это относится к системе обработки XML) библиотеки GNOME используются утилитами командной строки и серверными приложениями.

Конечно, нас более всего интересуют основы настольной среды и связанные с ней приложения. В последующих разделах мы коснемся внешнего вида GNOME, поговорим о существующих в этой среде возможностях настройки внешнего вида, а затем кратко познакомимся с основными приложениями, такими как Evolution и Nautilus.

Окружение рабочего стола GNOME входит в состав большинства дистрибутивов Linux. При его отсутствии или при желании иметь более свежую версию можно посетить сайт *http://gnome.org*[1] или веб-сайт конкретного дистрибутива, откуда скачать последнюю доступную версию пакета.

Базовый интерфейс рабочего стола

Рабочий стол GNOME спроектирован так, чтобы быть понятным всякому, кто ранее имел дело с компьютерами. Хотя можно изменить практически любые настройки, при типичной установке появляются рабочий стол со значками и двумя панелями вдоль верхнего и нижнего краев экрана. Панели принадлежат к наиболее важным инструментам GNOME, поскольку они универсальны и предоставляют многообразные возможности взаимодействия с системой. Панели могут располагаться вдоль всего края экрана, как панель задач Windows, вдоль части его, как Macintosh Dock, и массой других способов. Панели могут содержать меню, кнопки и маленькие приложения – *апплеты*, например часы, системные мониторы и даже маленькие игры.

Выполнение типичных действий в GNOME

Ниже следует краткое описание того, как выполнять стандартные задачи. Освоившись с ними, вы, вероятно, догадаетесь, как делать все остальное.

Открыть или активизировать элемент на панели

Щелкнуть на элементе левой кнопкой мыши один раз.

Запустить программу

Запуск программы производится щелчком левой кнопки мыши на *запускающем объекте*. В GNOME эти объекты обычно располагаются на обеих пане-

[1] Сайты *http://gnome.ru/* и *http://gnome.org.ru/* для русскоязычного читателя. – *Примеч. науч. ред.*

лях и на поверхности самого рабочего стола. Кроме того, когда производится щелчок на файле, он открывается ассоциированной с ним программой, о чем вскоре будет сказано дополнительно.

Переместить элемент по рабочему столу

Перетащить левой кнопкой мыши.

Переместить элемент по панели

Перетаскивание левой кнопкой мыши действует для запускающих объектов, но в некоторых апплетах левая кнопка используется для управления апплетом. В таком случае перетаскивайте средней кнопкой. То же касается перемещения окон захватом границы: щелчок левой кнопкой раскрывает окно, щелчок средней кнопкой перемещает его.

Упорядочить элементы на рабочем столе

Щелкнуть правой кнопкой на фоне рабочего стола и выбрать пункт контекстного меню Clean Up by Name (Выстроить по имени). Элементы будут отсортированы в алфавитном порядке с двумя исключениями: первый элемент в левом верхнем углу всегда будет ярлыком домашнего каталога пользователя, а последний элемент в списке – всегда мусорная корзина.

Открыть или активизировать элемент на рабочем столе

Дважды щелкнуть на элементе. Двойной щелчок на значке папки открывает ее в файловом менеджере Nautilus. Если дважды щелкнуть на документе электронной таблицы, запустится приложение электронной таблицы Gnumeric, в котором откроется документ. Двойной щелчок средней кнопкой мыши или щелчок с нажатой клавишей Shift на папке внутри открытого окна файлового менеджера приводит к закрытию текущего окна и открытию окна с содержимым выбранной папки.

Получить список параметров или задать свойства любого объекта

Щелкнуть правой кнопкой мыши, чтобы получить список параметров для любого объекта. Например, можно изменить фон рабочего стола, щелкнув на нем правой кнопкой и выбрав пункт контекстного меню Change Desktop Background (Изменить фон рабочего стола). Другие общие настройки можно выполнить с помощью центра управления GNOME, который вызывается через меню System (Система)→Settings (Параметры) или вводом команды `gnome-control-center` в командной строке. Точное содержание меню очень сильно зависит от дистрибутива и версии GNOME.

Вставить текст в любую текстовую область

Как и в любой другой операционной системе, копирование выполняется с помощью быстрой комбинации клавиш Ctrl+C, вырезание – Ctrl+X, и вставка – Ctrl+V. Исключение составляют Emacs и XChat. Можно использовать и более традиционный для UNIX способ, при котором вставка выделенного текста производится щелчком средней кнопки мыши.

Панель

Первоначальная конфигурация во многих системах задает узкую панель вверху и внизу экрана. На верхней панели есть ряд меню в левой части, несколько кно-

пок и часы – в правой. Нижняя панель содержит апплет списка окон, который должен показаться знакомым пользователям Microsoft Windows, – он отображает список всех открытых окон и позволяет быстро переключаться между ними.

Чтобы создать новую панель, щелкните правой кнопкой мыши по свободному месту на имеющейся панели и выберите пункт контекстного меню Create New Panel (Создать панель). Чтобы изменить свойства панели, например размер и цвет, щелкните на ней правой кнопкой и выберите пункт Properties (Свойства). Поэкспериментируйте с разными видами панелей и различными размерами, чтобы узнать, какие вам нравятся больше всего. Если у вас маленький экран типа лэптопа, скорее всего, вам подойдет меньший размер панели, чем при наличии большого экрана.

Чтобы добавить объекты запуска приложений на панель, можно перетащить их из меню или щелкнуть на панели правой кнопкой, выбрать пункт контекстного меню Add to Panel (Добавить на панель) и в открывшемся диалоге выбрать имя запускаемого приложения и значок для него. Можно также выбрать описание объекта, которое будет выводиться во всплывающей подсказке при наведении указателя мыши на значок объекта. Если вы хотите запускать приложение из терминала, установите параметр Run in Terminal (Приложение в терминале).

Для получения дополнительной информации о панели щелкните правой кнопкой по свободному месту на ней и выберите пункт контекстного меню Panel Manual (О панелях).

Апплеты панели – это маленькие приложения, выполняемые внутри панели. Их можно поместить на панель через меню Add to Panel (Добавить на панель) или просто запустить, выбрав требуемый из меню Applications (Приложения)→Applets (Апплеты). Разновидностей апплетов великое множество – от игр до утилит. Ниже приводится список некоторых из наиболее часто использующихся:

Область уведомлений

Область уведомлений напоминает системный лоток Windows и служит для размещения различных значков, отображающих состояние системы. Такие приложения, как Gaim – инструмент для быстрого обмена сообщениями (описывается в главе 5 «Веб-броузеры и обмен мгновенными сообщениями»), и Rhythmbox – музыкальный проигрыватель, используют область уведомлений с целью дать пользователям возможность обращаться к приложениям, не сохраняя при этом их окна открытыми. Системные предупреждения и предупреждения очереди печати также отображаются в этой области. И KDE, и GNOME используют одни и те же системные механизмы для создания системного лотка, поэтому апплеты, которые его используют, будут работать в любом из этих двух рабочих столов.

Монитор сети

Монитор сети запускается в области уведомлений и позволяет просматривать наличие подключений к сети и выбирать из них необходимые. Этот апплет особенно востребован пользователями ноутбуков, которые пользуются подключениями к сети типа Wi-Fi (802.11x). Для корректной работы монитора сети необходимо, чтобы был запущен демон netdaemon.

Системный монитор

Отображает график потребления системных ресурсов за последние несколько секунд. Чтобы получить более подробный отчет, включая список запущенных процессов и приложений, нужно щелкнуть на апплете правой кнопкой мыши и выбрать пункт контекстного меню Open System Monitor (Открыть системный монитор).

Переключатель рабочих мест

В большинстве установок этот апплет запускается при входе в систему и обычно настроен на четыре рабочих пространства. Каждое рабочее пространство служит эквивалентом нового экрана рабочего стола; их можно открыть в любом количестве. Переключатель рабочих пространств показывает все созданные вами виртуальные рабочие пространства и выводит каждое окно на рабочем столе в виде крошечного прямоугольника. С помощью левой кнопки мыши можно перетаскивать окна из одного рабочего пространства в другое. Чтобы изменить количество и расположение рабочих пространств, щелкните правой кнопкой и выберите пункт меню Properties (Свойства).

Список окон

Как и апплет рабочих пространств, список окон включается в большинство конфигураций. Он отображает открытые вами окна, чтобы можно было легко переходить из одного в другое, даже когда они свернуты. Если у некоторого приложения несколько окон, они группируются под одной записью. Чтобы отключить эту функцию или установить другие параметры апплета, щелкните правой кнопкой на апплете и выберите в контекстном меню пункт Properties (Свойства).

Индикатор состояния батареи

Индикатор состояния батареи показывает, сколько времени осталось работать источнику питания в портативных системах. Этот апплет может использоваться для приостановки системы: щелкните правой кнопкой мыши на апплете и выберите пункт меню Suspend Computer (Приостановить компьютер). Возобновление работы системы в этом случае происходит значительно быстрее, чем в случае обычной загрузки, но наличие такой возможности очень сильно зависит от имеющегося аппаратного окружения и самого дистрибутива. В старых системах, где использовалась расширенная система управления питанием (APM), для перехода в спящий режим использовалась команда *apm -s*. В современных системах, обладающих поддержкой ACPI, необходимо предварительно выполнить настройку всех событий ACPI в */etc/acpi/events/default*, хотя в современных дистрибутивах для выполнения подобных настроек имеются приложения с графическим интерфейсом. При наличии поддержки как ACPI, так и APM SUSE Linux для управления питанием использует демон *powersaved*, а переход в спящий режим производится командой *powersave --suspend*.

Nautilus: менеджер рабочего стола и файлов

Nautilus – это название менеджера рабочего стола и файлов в GNOME. Он управляет отображением фонового рисунка и файлов на рабочем столе, позволяет обрабатывать файлы без участия терминала и следит за мусорной корзиной. Ины-

ми словами, для GNOME это эквивалент Проводника Windows (Windows Explorer), Macintosh Finder и Konqueror из KDE. Подобно всем этим приложениям Nautilus позволяет перетаскивать элементы из одного места в другое. С его помощью можно просто копировать файлы комбинацией клавиш Ctrl+C, вырезать их комбинацией Ctrl+X и вставлять с помощью Ctrl+V.

В большинстве случаев Nautilus запускается сразу же после регистрации в системе. Если автоматический запуск Nautilus нежелателен, можно удалить эту программу из настроек сеанса с помощью утилиты Session Properties (Свойства сеанса) в центре управления GNOME. В последнем случае программу всегда можно будет запустить командой *nautilus*.

Быстрее всего начать работу с Nautilus можно, дважды щелкнув на значке домашнего каталога в левом верхнем углу рабочего стола, поименованного как домашняя папка пользователя. При этом откроется ваш домашний каталог. Nautilus отличается от других систем управления рабочим столом тем, что окно не только отображает содержимое каталога, но и само является каталогом: если открыть окно с некоторым каталогом, а затем попытаться открыть этот же каталог повторно, то это приведет к активации ранее открытого окна. По этой причине в верхней части окна отсутствует строка ввода адреса. Если необходимо сменить каталог, следует нажать комбинацию клавиш Ctrl+L и ввести новый адрес.

Опытные пользователи и те, кто знаком с другими системами управления рабочим столом, по достоинству оценят Nautilus, который, несмотря на кажущуюся простоту, обладает различными удобствами и быстрыми комбинациями клавиш, упрощающими и ускоряющими работу с программой. Первая из таких комбинаций – это Ctrl+L, которая одинаково работает не только в Nautilus, но и во всех диалогах выбора файлов рабочего стола GNOME, что позволяет отказаться от работы с мышью и вводить имена файлов вручную. Эта же комбинация клавиш позволяет вручную вводить адрес страницы в веб-броузерах.

Открытие окон: чтобы избежать загромождения рабочего стола открытыми окнами, можно открывать их с помощью левой кнопки мыши, одновременно удерживая клавишу Shift, или с помощью двойного щелчка средней кнопки мыши. В этом случае текущее окно будет закрыто до того, как откроется новое окно.

Комбинации быстрых клавиш для навигации от текущего местоположения: комбинация Alt+«стрелка вверх» открывает окно каталога, расположенного выше текущего в иерархии файловой системы, а Alt+Home открывает домашний каталог пользователя.

Если предпочтителен более подробный режим просмотра содержимого каталогов, достаточно щелкнуть правой кнопкой мыши на требуемом каталоге и выбрать пункт контекстного меню Browse Folder (Просмотреть папку). В подробном режиме окно программы включает в себя строку адреса, отсутствующую в обычном режиме, и дополнительную панель в левой части окна. В верхней части панели находится элемент выбора типа отображаемой информации:

Сведения

Отображает основные сведения о текущем каталоге.

Эмблемы

Отображает список доступных эмблем – маленьких значков, которые можно добавить к ярлыку каждого из файлов. Для этого достаточно лишь перетащить мышью нужную эмблему из панели к требуемому файлу. Например, если в каталоге хранится несколько однотипных файлов с похожими изображениями, можно перетащить эмблему Favorite (Избранное) или Cool (Круто) на один из них, чтобы впоследствии нужный файл можно было сразу же выделить при беглом просмотре. Эмблемы также можно устанавливать, выбрав пункт меню Edit (Правка)→Background and Emblems (Фон и эмблемы).

История

В этом режиме в панели отображается список каталогов, посещенных ранее с помощью Nautilus. Двойной щелчок на любом из каталогов вызывает переход в этот каталог.

Заметки

Позволяет оставить короткие заметки к какому-либо каталогу. Каждый каталог имеет свою страницу с заметками.

Дерево

Вероятно, один из самых полезных инструментов дополнительной панели программы Nautilus. Он позволяет перемещаться по дереву каталогов с помощью небольших изображений в виде треугольников. Рядом с каждым каталогом отображается маленький значок в виде треугольника, щелчком на котором можно открыть каталог и увидеть его содержимое без перехода в него.

У Nautilus есть ряд интересных дополнительных функций, включая следующие:

- Вместо общих значков для графических файлов Nautilus использует уменьшенные картинки самих изображений. Благодаря этому легко организовывать каталоги графических образов, например фотографий, полученных цифровой камерой.
- При наведении указателя мыши на музыкальный файл осуществляется его воспроизведение.
- Для текстовых файлов значок простого документа украшается фактическим содержимым файла. Благодаря этому можно вспомнить содержимое файла, не открывая его, даже если имя файла не слишком содержательно.
- Значок файла можно растянуть, если щелкнуть на нем правой кнопкой и выбрать пункт Stretch Icon (Растянуть значок). Если растянуть значок текстового файла в достаточной мере, можно увидеть целиком его содержимое и использовать как настольный блокнот.
- Выбрав пункт меню Edit (Правка)→Backgrounds and Emblems (Фон и эмблемы), можно выбирать различные эмблемы для декорирования ярлыков. Можно также перетаскивать образцы цвета и орнаменты для изменения оформления фона рабочего стола и панели. Чтобы изменить фоновый рисунок рабочего стола, необходимо щелкнуть правой кнопкой мыши на пустом пространстве рабочего стола и выбрать пункт контекстного меню Change Desktop Background (Изменить фон рабочего стола).

В целом Nautilus представляет собой универсальный инструмент, которым можно научиться пользоваться в результате небольшой практики. Дополнительную подсказку можно получить, выбрав пункт меню Help (Справка)→Nautilus User Manual (Содержание) в любом окне Nautilus.

Настройка среды GNOME для опытных пользователей: GConf

GConf – это централизованная система хранения настроек приложений рабочего стола, основанная на XML. Она позволяет приложениям совместно использовать настройки комбинаций быстрых клавиш, тем и прочих личных предпочтений и с помощью демона извещает запущенные приложения об изменении настроек, когда это происходит. Таким образом, отпадает необходимость перезапускать приложения, чтобы изменения вступили в силу.

Кроме того, GConf может использоваться для блокировки настроек рабочего стола с гораздо более высокой степенью избирательности, чем это возможно с традиционными блокировками файлов в UNIX. Администратор может открыть или закрыть доступ к тем или иным функциям некоторого приложения. Администраторы интерактивных терминалов, общественных компьютерных центров и других систем, где особое значение приобретает высокая степень защиты, понимают необходимость наложения определенных ограничений. Поэтому большинство приложений предоставляют возможность ограничения функциональности через свои файлы настроек GConf. Если у вас есть пользователи и вы хотите уберечь их от неприятностей, изучите эту тему глубже. Для этого рекомендуется прочитать прекрасное руководство «GNOME System Administrator's Guide», которое можно найти на сайте *http://www.gnome.org*.

В этой книге предполагается, что вопрос наложения ограничений на функциональные возможности не интересует читателя, тем не менее вы должны знать об этом, чтобы уметь выполнять их настройки по своему усмотрению. Для этих целей можно использовать *gconf-editor*. Разумеется, редактировать файлы в *~/.gconf* можно и вручную, но приложение *gconf-editor* сделает это занятие более простым и удобным.

Чтобы начать работу с приложением, запустите его командой *gconf-editor*. В левой части окна редактора отображается дерево настроек GConf, напоминающее дерево каталогов файловой системы и точно так же начинающееся с корня (/). Дерево параметров в точности соответствует содержимому файлов с настройками, находящимися в каталоге *~/.gconf*, таким образом, изменения в ветке */applications* будут отражаться на содержимом файлов, хранящихся в каталоге *~/.gconf/applications*. В правой части окна находится список доступных для настройки параметров, которые называются *ключами*, и место, где выводится описание выбранного ключа.

Основной интерес для нас представляют элементы в ветке */apps*. В ветках */desktop* и */GNOME* хранится информация, которая не имеет отношения к конкретным приложениям, как, например, данные сеанса и параметры настройки всего рабочего стола. В ветке */system* хранится информация общесистемного назначения, а информация о том, каким образом Gconf должен сохранять настройки, – в ветке */schemas*.

Теперь можно попробовать выполнить настройку какого-либо приложения, чтобы разобраться с тем, как это делается. Обычно файлы, находящиеся на рабочем столе, помещаются в каталог *~/Desktop*. Однако можно настроить Nautilus так, чтобы вместо этого он отображал содержимое домашнего каталога. Для этого нужно выбрать ключ */apps/nautilus/preferences/desktop_is_home_dir* и установить флажок. Теперь Nautilus будет отображать содержимое домашнего каталога на рабочем столе.

Другие приложения также имеют аналогичные «скрытые» настройки, которые доступны для изменения. Попробуйте сделать следующее:

- В настройках оконного менеджера Metacity: установить флажок */apps/metacity/reduced_resources*, чтобы уменьшить потребление ресурсов системы настолько, насколько это возможно. Это ухудшит внешний вид рабочего стола, но положительно скажется на производительности системы в целом.
- В настройках веб-броузера Epiphany: обычно щелчок средней кнопкой мыши в окне броузера Epiphany включает вертикальную прокрутку, эта особенность хорошо знакома пользователям Internet Explorer. Однако пользователи броузеров, традиционных для UNIX, предпочтут установить флажок */apps/epiphany/general/middle_click_open_url* и включить функцию вставки URL. Установив флажок, попробуйте выделить строку, содержащую URL, в любом из приложений и щелкнуть средней кнопкой мыши в окне броузера – в результате Epiphany загрузит выделенный текст.

Приложения GNOME

Освоившись с рабочим столом и основными операциями на нем, рассмотрим некоторые приложения, созданные для работы с ним. Обратите внимание, что использование этих приложений не ограничивается рабочим столом GNOME и это не единственные приложения, которые можно выполнять на рабочем столе GNOME, – просто они построены из одних и тех же материалов и исключительно хорошо взаимодействуют друг с другом.

Evolution: почта, календарь и контакты

Evolution – это то, что называют приложениями для рабочих групп (groupware suite): здесь объединены электронная почта, календарь и адресная книга, благодаря чему связь и планирование задач оказываются в одном удобном пакете. У нас нет места, чтобы подробно разбирать все три компонента, но есть полное руководство, включенное в меню Help (Справка) и доступное в Сети на сайте *http://gnome.org/projects/evolution*.

Запустить Evolution можно через соответствующий пункт в меню Applications (Приложения) либо путем ввода `evolution` в командной строке. При этом должен появиться экран, подобный показанному на рис. 3.7.

При первом запуске Evolution программа просит создать учетную запись электронной почты, для чего нужно ввести информацию о себе и доступе по электронной почте. Эти сведения можно взять из имеющейся почтовой программы либо узнать у системного администратора или у поставщика услуг подключения к Интернету.

Рис. 3.7. *Evolution на рабочем столе GNOME*

Evolution работает со стандартными протоколами почтовых серверов и может действовать практически в любом сетевом окружении. Пакет позволяет хранить почту на сервере (если взаимодействие с сервером осуществляется по протоколу IMAP), загружать в локальную систему (если взаимодействие с сервером осуществляется по протоколам IMAP или POP) или пользоваться спулингом почты на локальной системе, если у вас работает собственный почтовый сервер. Кроме того, Evolution поддерживает возможность взаимодействия с серверами Microsoft Exchange 2000 и Novell GroupWise 6.5, предоставляющих услуги электронной почты, календаря и адресной книги.

После создания учетной записи на экране появится основное окно Evolution. С левой стороны окна находится боковая панель быстрого доступа со списком доступных инструментов внизу и источников данных – вверху. Щелчком на одной из кнопок в нижней части панели производится переход к чтению электронной почты, календарю, списку задач, контактам и инструментам Microsoft Exchange.

Следующие разделы описывают три основные функции Evolution.

Электронная почта Evolution

Чтобы начать работать с почтой Evolution, щелкните на кнопке Inbox (Входящие) или выберите любую почтовую папку в панели папок. Окно представления почты делится на две части: в верхней располагается список сообщений, а в нижней находится область чтения выбранного сообщения. Можно изменять соотношение

частей, перетаскивая разделяющую их серую полосу, либо вообще скрыть область просмотра, выбрав пункт меню View (Вид)→Preview Pane (Панель предварительного просмотра) или нажав комбинацию клавиш Ctrl+'.

В целом функции почты довольно просты: чтобы отправить почту, помещенную в очередь для отложенной отправки, и проверить поступление новой, нужно щелкнуть на кнопке Send and Receive (Отправить/получить), а чтобы составить новое сообщение, щелкните на кнопке New Message (Создать).

Что отличает Evolution от других почтовых программ, так это скорость поиска, мощь и простота фильтров и уникальная характеристика vFolders – своего рода комбинация поиска с фильтрами.

Панель поиска располагается вверху списка сообщений. Для поиска в почте перейдите в любую почтовую папку, выделите ту часть сообщения, в которой нужно провести поиск (тело сообщения, отправителя, все сообщение и т. д.), введите в текстовое окно слово и нажмите клавишу Enter. Evolution выполняет предварительное индексирование почты, поэтому результат возвращается быстрее, чем в других программах.

Фильтры добавляют в конце поиска операцию: при каждом получении почты Evolution осуществляет в новых сообщениях заданный поиск и выполняет действия, зависящие от его результатов. Чаще всего фильтры применяют для автоматического размещения сообщений в зависимости от отправителей и удаления сообщений, помеченных как спам.

Чтобы создать фильтр, перейдите в любое представление почты и откройте список фильтров, выбрав пункт меню Tools (Сервис)→Filters (Настроить фильтры). Затем сделайте следующее:

1. Щелкните на кнопке Add (Добавить), чтобы добавить фильтр.
2. В верхней части окна выберите группу критериев отбора сообщений для фильтра. Например, если выбрать в первых раскрывающихся списках пункты Sender (Отправитель) и Contains (Содержит) и ввести `gnome.org` в появившемся рядом текстовом окне, то фильтр будет применяться ко всем сообщениям, поступающим с почтовых адресов *gnome.org*.
3. В нижней части окна выберите одно или несколько действий, выполняемых над сообщениями. Например, если выбрать Move to Folder (Переместить в папку), появится кнопка с надписью Click to Select Folder (Щелкните здесь, чтобы выбрать папку). Щелкните на ней, и вы сможете выбрать папку, в которую будет помещаться вся почта с адресов *gnome.org*.
4. Щелкните на кнопке OK в окне создания фильтра и на кнопке OK в списке фильтров. Создание фильтра закончено.

Если вам покажется недостаточной гибкость, предоставляемая фильтрами, можете воспользоваться vFolders. vFolder, или виртуальная папка, по существу, представляет собой результат сложного поиска, выглядящий как папка. Это означает, что, хотя почтовое сообщение может располагаться только в одной обычной папке, оно может находиться в нескольких виртуальных папках vFolder.

При создании виртуальной папки устанавливается критерий отбора, так же как это делается для фильтра, только вместо указания действий над сообщениями указывается место, где они должны храниться. Созданная виртуальная папка

появляется в списке виртуальных папок в нижней части дерева папок. Каждый раз, когда вы открываете такую папку, в почтовых папках выполняется поиск сообщений, отвечающих заданному при создании папки критерию. Таким образом, если вы создаете фильтры для размещения почты в зависимости от отправителя, то можете создать виртуальную папку, которая содержит сообщения с заданной темой вне зависимости от того, кто их отправил.

Работа с серверами GroupWise и Exchange осуществляется аналогичным образом, с единственным исключением. При работе с серверами GroupWise извещения доставляются не в папку входящих сообщений, а сразу в папку календаря. Как только контакт будет принят, он появится в вашем календаре. При работе с сервером Exchange ваше дерево каталогов содержит общедоступные папки. Чтобы получить доступ к общедоступным папкам других пользователей, нужно щелкнуть на кнопке Exchange и выбрать в меню Actions (Действия) пункт Subscribe to Other User's Folder.

Календарь Evolution

Календарь Evolution предоставляет большую гибкость в создании и просмотре расписаний. Чтобы начать с ним работать, щелкните на кнопке Calendar (Календарь) в панели быстрого доступа. Вам будет представлено пустое расписание на неделю без каких-либо назначенных встреч. В левой части окна выводится список имеющихся календарей, а с правой – содержимое выбранного календаря. Скрывать или показывать события, имеющиеся в отдельных календарях, можно, снимая или устанавливая флажок рядом с названием календаря в левой части окна. Каждый набор событий выделяется своим цветом, чтобы избежать путаницы, что позволяет упростить восприятие списков событий в процессе их сравнения, когда необходимо скоординировать их или ликвидировать возникающие конфликты.

Календари могут относиться к нескольким категориям: On this Computer (На этом компьютере), On the Web (В сети), Contacts (Контакты) и, в зависимости от типа сервера для рабочих групп, категории Exchange или GroupWise. В самом начале у вас имеется всего два календаря. По умолчанию первый из них – это ваш личный календарь. Второй – Birthdays and Anniversaries (Дни рождения) – содержит даты, которые были введены вами при добавлении контактов в адресную книгу.

Чтобы создать новый календарь, выберите пункт меню File (Файл)→New (Создать) →Calendar (Календарь) и в открывшемся диалоге укажите категорию создаваемого календаря: On this Computer (На этом компьютере) или On the Web (В сети). В случае выбора первой категории необходимо лишь определить имя нового календаря и его цвет, после чего можно щелкнуть на кнопке OK. Для календаря в сети потребуется ввести те же данные плюс URL файла календаря и частоту, с которой программа Evolution будет проверять наличие изменений.

Календари категорий GroupWise и Contacts (Контакты) создаются автоматически, и вы можете иметь только по одному календарю из этих категорий. Чтобы создать новый календарь из категории Exchange, необходимо с помощью инструмента Exchange подписаться на доступ к папке календаря на сервере Exchange.

Чтобы увидеть меньший или больший промежуток времени, можно выбрать диапазон дат в календаре, находящемся в правом верхнем углу, или щелкнуть

на панели инструментов на одном из предустановленных диапазонов дат: сегодня, одни сутки, пять дней, неделя или месяц.

Освоившись с перемещением по своему календарю, можно начинать планировать события. Чтобы создать событие, щелкните на кнопке New Appointment (Создать). Укажите название календаря и введите краткое описание события, выберите время и дайте (если хотите) более подробное описание. Обратите внимание: добавлять новые события можно не во все календари, так, календари в сети и календарь Contacts (Контакты) доступны только для чтения.

В нижнем правом углу расположен список категорий, к которым можно отнести событие. События, принадлежащие к одной из категорий, периодические события и напоминания отображаются в календаре с помощью маленьких значков: будильник для напоминаний, зацикленные стрелки для периодических событий, торт для дней рождения и т. д.

Можно также планировать напоминания и периодические события. Например, если на следующую неделю у вас назначена важная встреча, можно запланировать напоминание, которое всплывет за 15 минут до начала, чтобы вы могли приготовиться. Для этого щелкните на вкладке Reminder (Напоминания) и выберите время и тип напоминания, а затем щелкните на кнопке Add (Добавить), чтобы добавить его к списку. С периодическими событиями процедура аналогична: щелкните на вкладке Recurrence (Повторение) и выберите, как часто должно повторяться событие. Будет ли это только в четверг на этой неделе и во вторник на следующей? Будет ли оно повторяться каждую среду с сегодняшнего дня и до Рождества? Или это праздник, случающийся раз в год? Выберите правила повторения, щелкните на кнопке Save and Close (Сохранить и закрыть), и событие будет помещено в ваш календарь.

Осталось лишь скоординировать данное событие с другими людьми: щелкните на нем правой кнопкой мыши и выберите пункт контекстного меню Forward as iCalendar (Переслать как iCalendar), чтобы создать почтовое сообщение, к которому прикреплено событие. Когда адресат получит сообщение, ему достаточно будет один раз щелкнуть на кнопке, чтобы добавить событие в свой календарь и отправить вам сообщение, подтверждающее его согласие встретиться.

Адресная книга Evolution

Менеджер контактов Evolution, или адресная книга, представляет собой, вероятно, наименее яркую часть комплекта. Однако она тесно переплетена со средствами электронной почты. Карточки контактов создаются щелчком на кнопке New Contact (Создать) в представлении контактов, но можно создать карточку, щелкнув правой кнопкой на любом почтовом адресе в полученном сообщении электронной почты.

Если вы ввели информацию о днях рождения и годовщинах в карточки контактов, эти даты будут отображаться в специальном календаре.

Если вы ищете в адресной книге чей-то почтовый адрес, можете щелкнуть правой кнопкой на его карточке и выбрать отправку сообщения либо послать эту карточку кому-то другому, сделав два щелчка.

Чтобы посмотреть на менеджер контактов, щелкните на кнопке Contacts (Контакты) в панели быстрого доступа или выберите какую-нибудь папку с контактами в панели папок. Вы увидите простой список карточек. Если вы хотите упорядочить контакты, например список телефонов, выберите пункт меню View (Вид)→Current View (Показывать)→Phone List (Список телефонов). Можно также вывести список по организациям, а не по именам.

GNOME и офисное программное обеспечение

Окружение рабочего стола GNOME тесно интегрировано с пакетом офисных приложений OpenOffice, который дает возможность единообразной работы с текстовыми документами, электронными таблицами и презентациями. Пакет OpenOffice обладает превосходной совместимостью с файлами, созданными с помощью Microsoft Office, и предлагает богатый набор функциональных возможностей, необходимых для повседневного использования.

Однако существуют и другие офисные приложения. Например, приложение электронной таблицы Gnumeric умеет обращаться с некоторыми файлами, которые оказываются не под силу OpenOffice, и позволяет выполнять более сложные финансовые вычисления, хотя своими графическими возможностями оно не блещет. Приложение AbiWord – превосходный текстовый редактор, способный решать значительную часть задач и к тому же более простой, чем OpenOffice. Оба эти приложения занимают гораздо меньше дискового пространства, работают быстрее и подходят для установки в системы с ограниченными ресурсами.

Дополнительная информация об офисных пакетах приводится в главе 8.

Фильмы и музыка: Totem и Rhythmbox

Обсуждение возможности воспроизведения видео и музыки неизбежно сходится к обсуждению лицензирования. Группа, которая определила формат MP3, запатентовала алгоритмы кодирования и декодирования и требует, чтобы каждый поставщик отслеживал и оплачивал каждую копию программного обеспечения, проигрывающего или записывающего файлы в формате MP3, по этой причине не может существовать бесплатных и при этом юридически законных средств воспроизведения и записи MP3. Аналогичные ограничения накладывает DVD Copy Control Association (*dvdcca.org*), что не дает возможности разрабатывать бесплатные программные средства проигрывания DVD-фильмов, которые можно купить в магазине.

Нелицензионного программного обеспечения, способного воспроизводить MP3 и DVD, достаточно много, и любой сможет отыскать его с помощью поисковых машин, но в этом нет никакой необходимости. Можно записывать и прослушивать музыку в свободном формате Ogg Vorbis, а фильмы можно записывать и проигрывать в форматах MPEG и MOV, включая нешифрованный формат DVD, используемый бытовыми устройствами записи DVD.

Для воспроизведения музыки можно использовать аудиоплеер Rhythmbox, появившийся вслед за выходом проигрывателя iTunes для Apple. Проигрывателю Rhythmbox потребуется некоторое время, чтобы упорядочить коллекцию музыкальных файлов, прежде чем вы сможете приступить к прослушиванию. Если этого не произошло сразу же или программа не нашла все музыкальные файлы, выберите пункт меню Music (Музыка)→Import Folder (Добавить каталог).

После того как все файлы будут проиндексированы, перед вашим взором появится поразительно знакомый интерфейс: в левой части окна находится список источников музыки, включая пункты Library (Библиотека) и Radio (Радио), а также все списки воспроизведения, созданные вами. В правой части располагаются списки артистов и альбомов, которые могут использоваться для просмотра коллекции музыкальных произведений, а ниже всего этого находится список отдельных песен, которые соответствуют имени выбранного артиста и названию альбома. Кроме того, строка поиска, находящаяся в правом верхнем углу, позволяет производить поиск по артисту, альбому или названию песни.

Выберите песню и щелкните на кнопке Play (Воспроизвести). В процессе прослушивания можно щелкнуть правой кнопкой мыши на песне и выбрать пункт контекстного меню Properties (Свойства). На первой вкладке, Basic (Основные), содержатся лишь основные сведения о дорожке, зато на второй вкладке, Details (Детальные), выводятся сведения о том, как часто воспроизводится это произведение, где оно хранится, каков точный размер файла, а кроме того, здесь можно определить рейтинг произведения в диапазоне от 0 до 5. Если вы не собираетесь определять рейтинг произведения вручную, Rhythmbox предложит свои оценки, основанные на том, как часто это произведение воспроизводилось.

Еще одна из основных особенностей Rhythmbox – это списки воспроизведения. Чтобы создать собственный список воспроизведения, выберите пункт меню Music (Музыка)→Playlist (Список воспроизведения)→New Playlist (Создать список воспроизведения). Укажите название списка, после чего он появится в списке источников. Затем можно будет перетаскивать произведения из библиотеки во вновь созданный список воспроизведения.

Чтобы импортировать музыкальные произведения в Rhythmbox, необходимо иметь приложение, известное под названием Sound Juicer (Звуковыжималка), которое не всегда, но часто включается в состав дистрибутивов вместе с Rythmbox. Чтобы начать импорт произведений с компакт-диска, выберите пункт меню File→Import CD. Программа Sound Juicer проверит название компакт-диска, составит список дорожек, проверив их названия в Интернете с помощью службы MusicBrainz, и попросит вас подтвердить правильность сделанного, прежде чем импорт будет продолжен. После этого содержимое компакт-диска будет записано в виде файлов формата Ogg Vorbis, если не был выбран иной формат в диалоге настроек через меню Edit (Правка)→Preferences (Изменить настройки).

Воспроизведение видео с помощью программы Totem так же просто, как нажатие комбинации клавиш Ctrl+O, чтобы открыть файл (или Ctrl+L, чтобы открыть видеопоток в Web). Проигрыватель Totem предоставляет очень простой и понятный интерфейс к весьма сложному миру алгоритмов кодирования видео, однако спрятаться от потрясающего количества разных типов файлов удается далеко не всегда. Программа поддерживает по умолчанию несколько видеоформатов, включая те, что используются большинством видеокамер.

Вам не нужно монтировать DVD- или видеодиск: просто щелкните на кнопке воспроизведения. Однако при этом необходимо убедиться, что в системе присутствуют устройства */dev/dvd* или */media/dvd*.

Следует отметить, что видеоплеер Totem основан на использовании механизма воспроизведения Xine, который обладает такими же простыми настройками, что и Totem. Например, поддерживаются не все разновидности видеоформата QuickTime, однако пользователи современных компьютеров, построенных на базе микропроцессора x86, могут скопировать библиотеку QuickTime из Windows в каталог */usr/lib/win32* и получить доступ к поддержке этого видеоформата в своих системах. Кроме того, если была установлена программа RealPlayer for Linux, то видеоплеер Totem будет в состоянии воспроизводить файлы в формате RealVideo, используя для этого двоичные кодеки RealPlayer. Дополнительную информацию о воспроизведении мультимедиа в Linux, включая подсказки по улучшению производительности, обновления библиотек Xine и ссылки на другие программные продукты для воспроизведения мультимедиа, вы найдете на сайте *http:// www.xinehq.de*.

Дополнительные приложения и ресурсы

Существуют десятки, если не сотни, других приложений GNOME, включая средства программирования, игры, инструменты создания блок-схем и построения графиков. Узнать о них лучше всего, посетив веб-сайт *http://gnome.org* либо установив с помощью вашей системы обновления Red Carpet, *up2date*, *apt-get* или YaST.

Есть несколько мест, куда можно обратиться за помощью в случае возникновения трудностей. Помимо системы подсказки Nautilus и веб-сайта *gnome.org* можно поискать помощи в чатах. Разработчиков можно найти по адресу `irc.gnome.org` в канале `#gnome`, поэтому туда следует обратиться, если возникли вопросы по разработке программного обеспечения. Часто можно найти решение проблемы, если поискать в Сети текст сообщения об ошибке. Попытка поиска с помощью Google может привести вас на общественные форумы к людям, которые сталкивались с этой проблемой и, возможно, уже решили ее.

4

Основы командной строки UNIX

Если до Linux вы работали в Windows или другой операционной системе, отличной от UNIX, то вам предстоит многое изучить. Скажем откровенно, операционная система UNIX – это отдельный мир, но тем не менее за последние несколько лет работать с ней стало намного проще.[1]

В этой главе мы познакомим с основами UNIX тех читателей, которые никогда не сталкивались с этой операционной системой. Если вы раньше использовали Microsoft Windows или иную среду, то сведения из этой главы будут вам совершенно необходимы. В отличие от других операционных систем, UNIX вовсе не воспринимается интуитивно. Многие команды имеют странные имена или синтаксис, и причины этого закладывались в первые годы существования системы UNIX. И хотя многие команды выглядят похожими на свои аналоги в Windows[2], между ними существуют серьезные различия.

В этой главе мы стремимся избежать обсуждения вопросов обработки текста, синтаксиса языка командной оболочки и т. п. Вместо этого мы представим основные команды, которые позволят быстро освоить систему, если вы еще не знакомы со средой UNIX. Эта глава далека от полного изложения материала: настоящее введение в UNIX заняло бы целую книгу. Мы надеемся, что вы найдете в этой главе достаточно сведений для того, чтобы продолжить свое путешествие

---

[1] Операционные системы семейства Windows и семейства UNIX разрабатывались с совершенно разными целями. Хорошо известное утверждение гласит: «UNIX разрабатывалась программистами и для программистов»; в абсолютный противовес Windows разрабатывалась как система, ориентированная на конечного пользователя, который не знает (и, более того, не хочет знать) «кухню» ОС. Сравнивать или искать аналогии в этих ОС – все равно что сравнивать армейский внедорожник с седаном представительского класса – дело неблагодарное. Большая часть предметов, рассматриваемых авторами книги, как раз и относится к тому «тюнингу», который проделан «навстречу пользователю» в UNIX в последние 4–5 лет. – *Примеч. науч. ред.*

[2] Скорее, наоборот, можно говорить о командах Windows, имеющих аналоги в мире UNIX, так как именно из культуры UNIX заимствовались основные принципы и тенденции командного языка сначала MS DOS, а затем и Windows. – *Примеч. науч. ред.*

по Linux, и что в случае необходимости вы не будете скупиться и станете приобретать более специализированные книги. Мы дадим достаточную подготовку, чтобы вы смогли работать за терминалом, следить за выполнением заданий и вводить основные команды.

Во второй части книги содержится материал по администрированию и сопровождению системы. Это, несомненно, самая важная часть книги для каждого, кто работает в операционной системе Linux. Если вы ничего не знаете о UNIX, то после изучения данного начального руководства должны легко понять материал второй части книги.

Одна из серьезных тем, которой мы лишь коснемся в данной главе, – это редактирование файлов. Эта тема должна быть одной из первых при изучении любой операционной системы. О двух наиболее популярных в Linux редакторах, *vi* и Emacs, рассказано в главе 19.

Регистрация пользователя в системе

Допустим, что установка прошла гладко, и на экране появилось следующее приглашение:

```
Linux login:
```

Не всем пользователям Linux так везет: иногда приходится серьезно повозиться, чтобы вывести систему из сырого состояния или однопользовательского режима. Но сейчас мы говорим о регистрации в действующей Linux-системе.

Регистрация позволяет отличить одного пользователя от другого. Благодаря этому на машине могут работать одновременно несколько пользователей и исключается доступ посторонних к вашим файлам.

Возможно, вы установили Linux у себя дома и теперь думаете: «Большое дело. Все равно никто, кроме меня, системой не пользуется, и мне пока не нужно регистрироваться». Однако регистрация с личной учетной записью обеспечивает некоторую степень защиты: ваша учетная запись не позволит уничтожить или переместить важные системные файлы. Для таких деликатных[1] операций используется учетная запись администратора (о чем говорится в следующей главе).

Если вы подключаете свой компьютер к Интернету, пусть даже с помощью модема, все равно следите за тем, чтобы во всех учетных записях использовались нетривиальные пароли. Используйте в них специальные символы, знаки и строки, не являющиеся словами или именами. Несмотря на то, что операционная система UNIX менее подвержена различным атакам извне, чем Windows (согласно некоторым исследованиям компьютер, работающий под управлением операционной системы Windows, подвергается первой атаке уже через 20 минут после соединения с Интернетом, а чтобы загрузить исправления системы безопасности с сайта Microsoft, необходимо 40 минут), тем не менее это очень неприятно, когда кто-то заглядывает в ваши файлы.

---

[1] После «безобидной» команды от имени `rootchmod -R a-X /` систему, скорее всего, придется переустанавливать. – *Примеч. науч. ред.*

Следует учесть, что в некоторых дистрибутивах для регистрации сразу вызывается графический администратор регистрации, и вы не увидите таинственного приглашения `login:`, написанного белыми буквами на черном фоне. Вместо этого вам предложат красивый графический экран, возможно, со списком пользователей, у которых есть учетные записи на вашей машине (и иногда даже с маленькими изображениями для каждого из пользователей), и различных режимов регистрации. Тем не менее основная процедура регистрации остается той же самой: вы вводите свое имя и пароль.

Возможно, во время установки вам было предложено создать личную учетную запись. Если такая запись есть, введите выбранное вами имя в ответ на приглашение `Linux login:`. Если у вас еще нет учетной записи, введите `root`, поскольку эта учетная запись существует всегда. В некоторых дистрибутивах создается учетная запись пользователя с именем *install* или каким-либо другим, позволяющая зарегистрироваться сразу же после установки системы.

После того как будет введено имя пользователя, вы увидите приглашение

```
Password:
```

и должны будете ввести правильный пароль. Во время этой операции терминал отключает обычное отображение вводимых символов на экране, чтобы никто не мог увидеть ваш пароль, глядя на экран. Если такой подсказки не появилось, нужно установить пароль, чтобы защитить себя от постороннего вмешательства. О том, как это сделать, мы расскажем позднее.

Между прочим, как имя, так и пароль чувствительны к регистру.[1] Проверьте состояние клавиши `Caps Lock`, потому что `ROOT` и `root` – это разные имена.

После успешной регистрации вы увидите приглашение к вводу. Если вы зарегистрировались как *root*, это может быть просто:

```
#
```

Для других пользователей приглашением к вводу обычно является символ `$`.[2] В приглашении может также содержаться имя, присвоенное вами системе, или каталог, в котором вы в данное время находитесь. Что бы там ни появилось, теперь можно вводить команды. Мы говорим, что сейчас вы находитесь на «уровне командной оболочки», а приглашение, которое вы видите, – это «приглашение командной оболочки», поскольку вы сейчас работаете в программе, называемой *командной оболочкой* (*shell*), которая обрабатывает ваши команды. Пока мы можем не думать об оболочке, но позднее в данной главе мы продемонстрируем ряд ее полезных особенностей.

---

[1] Вообще, различие заглавных и прописных литер – общее свойство ОС семейства UNIX; наиболее отчетливо это выражается в написании имен файлов: *myfile* и *MyFile* – это два файла, не имеющие ничего общего. Поэтому проще один раз и навсегда запомнить, что в UNIX литеры `a` и `A` – две различные литеры, которые ничто не связывает, – и это будет «работать» во всех случаях. – *Примеч. науч. ред.*

[2] Вид «приглашения» – как в режиме *root*, так и в режиме пользователя – вещь перестраиваемая, так что вы можете позже перестроить их к виду, который вам больше нравится. – *Примеч. науч. ред.*

В этой главе при демонстрации команд мы будем изображать приглашение просто как $. Поэтому, если вы увидите:

```
$ pwd
```

это означает, символ $ выводится оболочкой, а вы должны ввести `pwd`.

Установка пароля

Если у вас еще нет пароля[1], советуем его установить. Просто введите команду *passwd*. Вам будет предложено ввести пароль, а затем повторить ввод, чтобы застраховаться от ошибок при вводе.

Есть стандартные правила выбора пароля, которые затрудняют возможность угадать его для посторонних. Некоторые системы даже проверяют ваш пароль и отклоняют его, если он не удовлетворяет некоторым минимальным требованиям. Например, часто требуется, чтобы пароль содержал не менее шести символов. Более того, пароль должен содержать символы верхнего и нижнего регистров или включать символы, отличные от букв и цифр.

Если вы думаете, что в качестве пароля вполне сгодится обычное слово, пусть и редко используемое, советуем вам подумать еще раз. Существуют программы подбора паролей, которые включают в себя словарь английского языка и просто перебирают все слова в словаре, пока не найдут нужное. Поэтому при использовании обычных слов в качестве паролей ваша учетная запись может быть достаточно легко скомпрометирована. Кроме того, старайтесь не использовать имя учетной записи в качестве пароля. Если имя учетной записи «joe», первое, что попробует сделать злоумышленник, это попытаться использовать имя учетной записи в качестве пароля.

Самый простой, но эффективный способ при выборе пароля заключается в том, чтобы подобрать достаточно длинную фразу, которую легко можно запомнить (например, строка из любимой песни), и составить пароль из первых букв слов этой фразы. Далее к этому паролю можно добавить цифры и какие-нибудь специальные символы. Рассмотрим в качестве примера фразу *I'd really like to go fishing now* (Я действительно не прочь отправиться сейчас на рыбалку), тогда пароль мог бы выглядеть как-то так: *Irl2gfn!*. Но не используйте пароль, который только что был предложен, поскольку все, что приводится в книге, не может рассматриваться в качестве кандидата на роль безопасного пароля. Существуют такие программы (они крайне редко включаются в состав графического инструмента регистрации пользователя), которые помогут сгенерировать случайный пароль из случайного набора символов. Разумеется, такие пароли достаточно сложно запоминать, а кроме того, если пароль приходится записывать куда-нибудь, чтобы не забыть его, – это тоже плохой пароль.

---

[1] Здесь есть различие в поведении разных ОС семейства UNIX и даже различных версий одной ОС: некоторые могут не допускать имени регистрации без пароля (с «пустым» паролем), другие могут не допускать отсутствия пароля для *root*, но допускать для остальных пользователей (что достаточно разумно), третьи могут допускать отсутствие пароля для всех. – *Примеч. науч. ред.*

Чтобы изменить пароль, снова введите команду `passwd`. Она предложит ввести старый пароль (чтобы убедиться, что вы действительно хозяин этой учетной записи), а затем позволит его изменить.[1]

Виртуальные консоли

Будучи многозадачной средой, Linux предоставляет ряд интересных способов одновременного выполнения нескольких задач. Вы можете начать долгую установку программного обеспечения, а затем переключиться на чтение электронной почты или компиляцию программы, и все это будет выполняться одновременно.

Большинство пользователей Linux, которым нужен такой асинхронный доступ, пользуются X Window System (глава 16). Но прежде чем запустить X, можно сделать нечто подобное с помощью виртуальных консолей. Такая возможность существует в некоторых других версиях UNIX, хотя не во всех.

Чтобы переключиться в другую виртуальную консоль, нажмите левую клавишу Alt и, удерживая ее нажатой, нажмите одну из функциональных клавиш от F1 до F8.[2] При нажатии каждой функциональной клавиши вы будете попадать в новый экран с приглашением к регистрации. Можно регистрироваться в различных виртуальных консолях, как если бы вы были разными людьми, и переключаться между консолями для выполнения различных действий.[3] Можно даже в каждой консоли запустить свой сеанс X. X Window System по умолчанию использует виртуальную консоль с номером 7. Поэтому, запустив X и переключившись затем в одну из текстовых виртуальных консолей, вы можете вернуться в X, нажав комбинацию Alt+F7. Если вы обнаружите, что нажатие комбинации Alt+<функциональная клавиша> вместо переключения виртуальных консолей вызывает меню X или некоторую другую функцию, используйте комбинацию Ctrl+Alt+<функциональная клавиша>. Существует возможность запустить два X-сервера, в этом случае вторая графическая консоль с X Window System будет иметь номер 8.

Часто используемые команды

Количество команд в типовой системе UNIX таково, что можно заполнить несколько сотен страниц справочного руководства. Кроме того, существует возможность добавлять новые команды. Команд, о которых мы вам расскажем, будет вполне достаточно, чтобы перемещаться по каталогам и смотреть, что имеется в системе.

---

[1] Если вы забыли пароль для *root*, можете считать, что вам сильно не повезло: скорее всего, вам просто придется переустанавливать систему «с нуля». – *Примеч. науч. ред.*

[2] Число консолей является конфигурируемым параметром системы. – *Примеч. науч. ред.*

[3] Существует и третий способ запустить приложения на параллельное исполнение, даже не покидая консоли, – фоновое исполнение команды. Для этого нужно вместо команды `# команда` набрать `# команда &`. – *Примеч. науч. ред.*

Каталоги

Так же как в Windows и практически любой современной компьютерной системе, файлы UNIX организованы в виде иерархической структуры каталогов.[1] UNIX не накладывает ограничений на то, где должны располагаться файлы, но за годы существования этой операционной системы возникли некоторые общепринятые соглашения. Так, например, в Linux вы найдете каталог с именем */home*, в который помещаются файлы всех пользователей. У каждого пользователя в */home* имеется свой подкаталог. Поэтому если ваше имя в системе – *mdw*, то ваши личные файлы будут располагаться в каталоге */home/mdw*. Этот каталог называется вашим *домашним* (*home*) *каталогом*. Конечно, вы можете создавать в нем другие подкаталоги.

Если вы работали раньше в Windows, то использование символа слэша (/) в качестве разделителя может показаться странным, поскольку вы привыкли к символу обратного слэша (\). В символе слэша нет ничего необычного, и он использовался в качестве разделителя задолго до того, как появились MS-DOS или Windows. Обратный слэш имеет в UNIX другое назначение (отменяет специальное значение следующего за ним символа, если таковое у него есть).[2]

Как видите, составляющие каталога разделяются символом слэша. Такой список, компоненты которого разделены символом слэша, часто называют *путем* (*pathname*).

В каком каталоге находится */home*? Разумеется, в каталоге с именем /. Он называется *корневым каталогом*. Мы уже говорили о нем при настройке файловых систем.

После регистрации система делает ваш домашний каталог текущим. Чтобы убедиться в этом, используйте команду «вывести рабочий каталог» (print working directory), или `pwd`:

```
$ pwd
/home/mdw
```

Система подтверждает, что вы находитесь в каталоге */home/mdw*.

Конечно, находиться все время в одном каталоге не очень весело. Попробуем использовать другую команду, `cd`, чтобы перейти в другой каталог:

```
$ cd /usr/bin
$ pwd
/usr/bin
$ cd
```

---

[1] Это немного «поспешное» утверждение: в Windows существуют *несколько* иерархических файловых систем, начинающихся от имени диска: *C:*, *D:* и т.д.; в UNIX, напротив, существует *единственная* однородная файловая система, начинающаяся от корня \, к которой отдельные «диски» и даже фрагменты файловых систем удаленных сетевых компьютеров подключаются в качестве составных частей единого целого так называемой операцией монтирования. – *Примеч. науч. ред.*

[2] Или перенос длинной командной строки на следующую строчку. – *Примеч. науч. ред.*

Где теперь мы находимся? Команда `cd` без аргументов возвращает нас в наш домашний каталог. Между прочим, домашний каталог часто обозначается символом «тильда» (~). Поэтому строка *~/programs* означает каталог *programs*, который находится в домашнем каталоге.

Раз уж на то пошло, давайте создадим каталог с именем *~/programs*. Находясь в домашнем каталоге, можно ввести либо:

```
$ mkdir programs
```

либо полный путь:

```
$ mkdir /home/mdw/programs
```

Теперь перейдем в новый каталог:

```
$ cd programs
$ pwd
/home/mdw/programs
```

Специальная последовательность символов «..» относится к каталогу, расположенному непосредственно выше текущего, или *родительскому*.[1] Поэтому в домашний каталог можно вернуться, если ввести следующее:

```
$ cd ..
```

Противоположностью `mkdir` служит команда `rmdir`, которая удаляет каталоги:

```
$ rmdir programs
```

Аналогично команда `rm` удаляет файлы. Мы не говорим о ней сейчас, поскольку не показали еще, как создавать файлы. Для этого обычно используются редакторы *vi* или Emacs (глава 19), но создавать файлы можно и с помощью некоторых команд, о которых будет рассказано далее в этой главе. Запущенная с ключом *-r* («recursive»), команда *rm* удалит весь каталог и его содержимое (будьте осторожны!).

Следует отметить, что графические окружения рабочих столов для Linux, такие как KDE и GNOME, включают в себя свои собственные менеджеры файлов, которые могут выполнять большинство операций[2], описанных в этой главе: вывод списка и удаление файлов, создание каталогов и т. п. Некоторые из них, как, например, Konqueror (входящий в состав KDE и одновременно исполняющий функции веб-броузера в этой среде), обладают весьма широкими возможностя-

---

[1] Так же как ../ является синонимом «родительского каталога», ./ является синонимом «текущего каталога». Зачем еще нужен синоним текущему каталогу, если мы и так в нем находимся? Вот ошибка, на которую попадался практически любой пользователь UNIX: запуск программы из текущего каталога, когда текущий каталог не включается в состав переменной окружения `PATH` (а в UNIX, в отличие от Windows, так и устанавливается): `# имя-программы` не сработает, а `# ./имя-программы` – сработает! – *Примеч. науч. ред.*

[2] Но при этом хорошо «держать в уме», что в UNIX такие оболочки просто за нас «складывают текст» командной строки, который потом передают оболочке на исполнение, в отличие от того, как это часто реализуется в Windows, например файловыми менеджерами NC, FAR и др. – *Примеч. науч. ред.*

ми. Однако, когда речь заходит о выполнении операций сразу над несколькими файлами, имена которых, возможно, следуют определенным правилам, очень сложно превзойти командную строку по эффективности, но даже это требует времени на обучение. Например, если необходимо удалить все файлы в текущем каталоге и всех вложенных подкаталогах, имена которых начинаются с *r* и заканчиваются на *.txt*, то в командной оболочке Z (*zsh*) сделать это можно одной командой::

```
$ rm **/r*.txt
```

Подробнее о подобных приемах работы в командной строке мы поговорим далее.

Вывод списка файлов

Чтобы просмотреть содержимое каталога, введите *ls*. Команда *ls* без параметров показывает содержимое текущего каталога. Добавив имя другого каталога в качестве аргумента, можно увидеть его содержимое:

```
$ ls /home
```

В некоторых системах[1] команда `ls` работает причудливо и отображает специальные файлы, такие как каталоги и исполняемые файлы, полужирным шрифтом или даже различными цветами. Если вы хотите использовать возможность выделения цветом или определить другие цвета, отредактируйте файл */etc/DIR_COLORS* или создайте в своем домашнем каталоге его копию с именем *.dir_colors* и отредактируйте ее.

Как и для большинства команд UNIX, можно управлять `ls` с помощью параметров[2], начинающихся с дефиса (`-`). Не забывайте вводить перед дефисом пробел. Полезным параметром команды *ls* является *-a* (all), который позволит вам увидеть в своем домашнем каталоге сокровища, о которых вы и не подозревали:

```
$ cd
$ ls -a
.                .bashrc              .fvwmrc
..               .emacs               .xinitrc
.bash_history    .exrc
```

Одна точка означает ссылку на текущий каталог, а две точки – на каталог, находящийся уровнем выше. Но что это за файлы, имена которых начинаются с точки? Они называются *скрытыми*. Точка перед именем предотвращает их отображение при выполнении обычной команды *ls*. Многие программы используют скрытые файлы для определения пользовательских параметров – описания их поведения по умолчанию, которое можно изменить. Например, в файл *.Xdefaults* можно поместить команды, которые изменяют характер выполнения программ, использующих X Window System. О существовании большинства из этих фай-

---

[1] Или более строго: в некоторых командных оболочках (любой системы). – *Примеч. науч. ред.*

[2] Модификатор, начинающийся с `-`, называется «switch» (ключ, переключатель), а уже после всех ключей команды могут следовать «параметры», которым «-» не предшествует, например `# ls -l /home`. Здесь `-l` – ключ, а `/home` – параметр. – *Примеч. науч. ред.*

лов можно даже не вспоминать, но при настройке системы они будут иметь для вас очень большое значение. Некоторые из них мы перечислим позже.

Другим полезным параметром *ls* является *-l* (long). Он позволяет вывести дополнительные сведения о файлах. На рис. 4.1 показан результат работы команды *ls* с описанием всех полей. Добавление параметра *-h* (human) приводит к тому, что размеры файлов округляются до величин, более удобочитаемых с точки зрения человека.

О полях Permissions (Права доступа), Owner (Владелец) и Group (Группа) мы расскажем в главе 11. Команда `ls` выводит также размер каждого файла и время его последнего изменения.

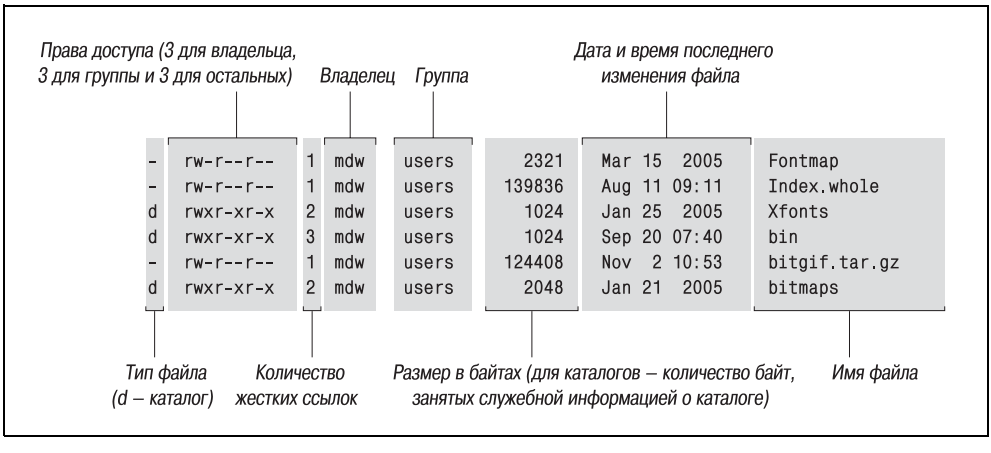


Рис. 4.1. *Результат выполнения команды ls -l*

Просмотр файлов, команды more и less

Один из способов увидеть содержимое файла – это вызвать редактор, например:

```
$ xemacs .bashrc
```

Однако, если вам нужно лишь просмотреть файл, а не редактировать его, это можно сделать быстрее с помощью других команд. Проще всего использовать команду со странным именем `cat` (от «concatenate», поскольку ее можно также использовать для конкатенации, то есть объединения нескольких файлов в один[1]):

```
$ cat .bashrc
```

Но длинный файл промелькнет слишком быстро, чтобы можно было просмотреть его содержимое, поэтому вместо `cat` чаще используется команда `more` («более»):

```
$ more .bashrc
```

Эта команда выводит часть содержимого файла, умещающуюся на целом экране, и ждет нажатия клавиши «пробел» для вывода очередной порции. У коман-

[1] `# cat doc1 doc2 doc3 > doc.all`. – *Примеч. науч. ред.*

ды more много самых разных параметров. Например, можно осуществлять поиск строки в файле: нажмите клавишу слэш (/), введите строку и нажмите Enter.

Одной из известных разновидностей команды more является команда *less* («менее»). Эта команда обладает еще большими возможностями. Например, она позволяет пометить в файле некоторое место и позднее вернуться к нему.

Символические ссылки

Иногда бывает удобно хранить файл в одном месте так, чтобы казалось, будто он находится в другом. Чаще всего это делает не пользователь, а системный администратор. Например, у вас может быть несколько версий некоторой программы с именами *prog.0.9*, *prog.1.1* и т. д., но обращаться к используемой в данное время версии нужно по имени *prog*.[1] Либо вы можете поместить файл в некоторый дисковый раздел, где есть свободное пространство, а использующей этот файл программе нужно, чтобы он находился в другом месте, поскольку путь к нему жестко зашит в программном коде.

Для таких случаев в UNIX предусмотрены *ссылки* (*links*). В этом разделе мы рассмотрим символические ссылки[2], являющиеся наиболее гибким и распространенным типом ссылок. Символическая ссылка – это в своем роде фиктивный файл, просто указывающий на другой файл. Если вы пытаетесь редактировать, читать или выполнять символическую ссылку, то система поймет, что вам нужна не сама ссылка, а реальный файл. Символические ссылки во многом похожи на ярлыки в Windows 95/98[3], но обладают значительно большими возможностями.

Вернемся к примеру с *prog*. Нужно создать ссылку с именем *prog*, указывающую на реальный файл с именем *prog.1.1*. Введите команду:

```
$ ln -s prog.1.1 prog
```

В результате будет создан новый файл с именем *prog*, являющийся в некотором роде фиктивным: при его запуске в действительности запустится *prog.1.1*. Посмотрим, что сообщит об этом файле команда ls -l:

```
$ ls -l prog
lrwxrwxrwx   2 mdw     users       8 Nov 17 14:35 prog -> prog.1.1
```

Символ l в начале строки показывает, что файл является ссылкой, а стрелка -> обозначает реальный файл, на который указывает ссылка.

Пользоваться символическими ссылками очень просто, если освоиться с мыслью о том, что один файл может указывать на другой. При установке программных пакетов приходится постоянно сталкиваться со ссылками.

---

[1] На этом построены разнообразные пакетные и инсталляционные подсистемы в различных UNIX-системах. – *Примеч. науч. ред.*

[2] Кроме символических ссылок в UNIX предусмотрены «жесткие» ссылки, появившиеся намного раньше символических; внешне они могут выглядеть очень похоже, разница при создании будет проявляться только в наличии (для символических) или отсутствии (для жестких) ключа -s в команде: # ln -s prog.1.1 prog. – *Примеч. науч. ред.*

[3] Ярлыки Windows появились на несколько десятилетий позже ссылок UNIX, и, скорее всего, «по мотивам». – *Примеч. науч. ред.*

Командные оболочки

Как уже говорилось ранее, после входа в систему в консольном режиме вы попадаете в командную оболочку. Если ваша система настроена так, чтобы использовать графический менеджер регистрации, то после входа в систему вы попадаете в графический интерфейс, в котором для получения доступа к оболочке можно открыть окно терминала *xterm* (или подобного ему). Оболочка интерпретирует и выполняет все ваши команды. Прежде чем продолжить, взглянем на различные имеющиеся оболочки, поскольку они связаны с частью последующего материала.

Разнообразие командных оболочек в UNIX может привести в замешательство кого угодно, но к этому нужно относиться как к результату эволюционного развития. Можете поверить, что вам не захотелось бы остановиться на самой первой разработанной для UNIX оболочке, оболочке Борна (Bourne shell). Хотя для своего времени (середина 70-х годов прошлого века) это был очень мощный интерфейс пользователя, тем не менее в нем отсутствовали многие полезные функции, необходимые для работы в интерактивном режиме, в том числе те, о которых рассказывается в этом разделе. Поэтому со временем были разработаны другие оболочки, и сегодня можно выбрать из них ту, которая более всего подходит вашему стилю работы.

Ниже перечислены некоторые командные оболочки, имеющиеся в Linux:

bash

Еще одна оболочка Борна (Bourne Again shell). Наиболее часто используемая и самая мощная оболочка в Linux. POSIX-совместимая и совместимая с оболочкой Борна, создана и распространяется в рамках проекта GNU (Фонд свободного программного обеспечения). Поддерживает функции редактирования командной строки, подстановку строк из истории и совместимость с оболочкой Борна.

csh

Оболочка C. Разработана в Беркли. Совместима с оболочкой Борна при работе в интерактивном режиме, но имеет совершенно другой командный язык. Не обеспечивает возможность редактирования командной строки, но имеет очень сложную альтернативу, называемую *подстановкой истории*. В Linux оболочка *csh* служит просто псевдонимом для более новой *tcsh*.

ksh

Оболочка Корна (Korn shell). Пожалуй, самая популярная в UNIX-системах и первая, в которой появились современные технологии, используемые в командных оболочках (в том числе заимствованные из оболочки C). Совместима с оболочкой Борна. Поддерживает возможность редактирования командной строки.

sh

Оболочка Борна (Bourne shell). Самая первая командная оболочка. Не поддерживает возможность редактирования командной строки.

tcsh

Расширенная оболочка C. Поддерживает возможность редактирования командной строки.

zsh

Оболочка Z. Самая новая из командных оболочек. Совместима с оболочкой Борна. Поддерживает возможность редактирования командной строки. Обладает широкими возможностями автодополнения. Если вы еще не знакомы ни с одной из командных оболочек и в вашем дистрибутиве имеется *zsh*, рекомендуем остановить свой выбор на этой оболочке.

Определить, какая оболочка запущена, можно с помощью следующей команды, которая покажет полный путь к месту нахождения оболочки (не пропустите знак доллара перед именем переменной окружения):

```
$ echo $SHELL
```

В большинстве наиболее популярных дистрибутивов в качестве командной оболочки используется *bash* (Bourne Again shell), поэтому, скорее всего, у вас работает оболочка *bash*. Если какая-нибудь другая, то самое время перейти на *bash* или *zsh*. Они обладают широчайшими возможностями, совместимы со стандартами POSIX, имеют неплохую поддержку и очень популярны в Linux. Чтобы сменить оболочку, выполните команду `chsh`:

```
$ chsh
Enter password: Здесь нужно ввести свой пароль - это в целях безопасности
Changing the login shell for mdw
Enter the new value, or press return for the default

    Login Shell [/bin/sh]:/bin/bash
```

(Для выбора командной оболочки *zsh* используйте путь */usr/bin/zsh* или */bin/zsh*, в зависимости от того, где находится исполняемый файл оболочки в вашем дистрибутиве.)

Чтобы пользователь мог выбрать некоторую оболочку, запускаемую после регистрации, ее нужно установить, а системный администратор должен сделать ее доступной, поместив в файл */etc/shells*.

Оболочки можно разбить на классы, пользуясь разными критериями. Во-первых, можно разделить оболочки на Борн-совместимые и *csh*-совместимые. Такое разделение представляет интерес, когда вы начинаете программировать на языке командной оболочки, то есть создавать сценарии. Оболочки Борна и оболочка C отличаются друг от друга языковыми конструкциями. По мнению большинства Борн-совместимые оболочки предпочтительнее, и многие утилиты UNIX признают только оболочку Борна.

Во-вторых, можно разделить оболочки на те, которые поддерживают возможность редактирования командной строки (все современные оболочки), и те, которые ее не поддерживают. В оболочках *sh* и *csh* эта удобная возможность отсутствует.

При объединении обоих критериев – совместимость с оболочкой Борна и возможность редактирования командной строки – лучшим выбором становятся *bash*, *ksh* и *zsh*. Попробуйте поработать с несколькими оболочками, прежде чем остановиться на одной из них. Это будет полезно на тот случай, если когда-нибудь вы столкнетесь с системой, где выбор оболочек будет ограничен.

Быстрые комбинации клавиш и как ими пользоваться

При вводе команды нажатие клавиши Backspace удаляет последний символ. Комбинация Ctrl+U удаляет строку от курсора до ее начала, поэтому при нахождении курсора в конце строки удаляется вся строка. Если вы закончили ввод команды и она начала выполняться, то при нажатии Ctrl+C ее выполнение должно быть прекращено, а при нажатии Ctrl+Z – приостановлено. (При желании возобновить выполнение приостановленной программы можно с помощью команды *fg* (от «foreground» – передний план).)

Комбинация Ctrl+S останавливает вывод на экран, пока он не будет продолжен с помощью Ctrl+Q. Сегодня это, возможно, не так полезно, поскольку в большинстве эмуляторов терминала существует возможность прокрутки экрана, но об этой комбинации нужно помнить на тот случай, если вы случайно нажмете Ctrl+S и терминал внезапно «замрет»; тогда нужно просто нажать Ctrl+Q, и он снова оживет.[1]

Если некоторые из этих комбинаций не работают, значит, в силу каких-то причин ваш терминал был неправильно настроен. Исправить это можно командой `stty`, используя следующий синтаксис:

```
stty function key
```

где *function* – это то, что вы хотите сделать, а *key* – клавиша, которую нужно нажать. Клавиша Ctrl обозначается диакритическим знаком «крышечка» (^) перед клавишей.

Ниже приводится группа команд для установки описанных выше функций:

```
$ stty erase ^H
$ stty kill ^U
$ stty intr ^C
$ stty susp ^Z
```

Первая клавиша, `^H`, представляет ASCII-код, генерируемый клавишей Backspace.

Кстати, список текущих установок терминала можно получить, введя команду `stty -a`. Но разобраться в результатах ее работы не так-то просто: *stty* – сложная команда, имеющая многочисленные применения, и для некоторых из них требуются обширные познания в терминалах.

---

[1] Здесь для упрощения изложения допущено существенное смешение понятий: если «горячая комбинация» Ctrl+U является командой непосредственно оболочке «удалить строку ввода», то все остальные комбинации, упоминаемые в предыдущих абзацах (Ctrl+C, Ctrl+Z и т. д.), – это указание *shell* отправить «сигнал POSIX» процессу, выполняемому на текущий момент под управлением *shell*. Оболочка передает этот сигнал транзитом, не анализируя то, что она передает. Реакция процесса-получателя на полученный сигнал может совпадать с описываемым в тексте, но может и радикально отличаться, если процесс переопределил реакцию на сигнал (исключение составляет очень небольшая группа сигналов, для которых нельзя переопределить реакцию, например SIGKILL, но для таких сигналов нельзя и установить комбинацию «горячих клавиш» *shell*). – *Примеч. науч. ред.*

Ниже приводится еще один пример использования stty. Если настройки терминала случайным образом были изменены (что часто случается при выводе двоичных данных) и он реагирует не так, как вы того ожидаете, попробуйте ввести:

```
$ sty sane
```

Обычно этот прием помогает «вразумить» терминал и заставить его работать должным образом.

Сокращенный ввод с клавиатуры

Если вы повторяете команды этого руководства, работая за терминалом, то, вероятно, уже устали неоднократно вводить одно и то же. Особенно неприятно сделать ошибку и начать все сначала. Здесь оболочка может облегчить жизнь. Она не упрощает UNIX до интерфейса «укажи и щелкни», но может помочь быстрой работе в командной среде.

В этом разделе рассказывается о редактировании командной строки. Эти советы помогут, если у вас оболочка *bash*, *ksh*, *tcsh* или *zsh*. Функция редактирования командной строки запоминает последние 50 строк (или около того) в виде буфера редактора (его еще называют *историей команд*). Можно перемещаться по этим строкам и изменять их так же, как при редактировании документа. Каждый раз, когда нажимается клавиша Enter, оболочка исполняет текущую строку и запоминает ее в истории команд.

Функция автодополнения

Сначала попробуем что-нибудь не очень сложное, но способное помочь сберечь массу времени. Введите следующие символы, не нажимая клавишу Enter:

```
$ cd /usr/inc
```

Теперь нажмите клавишу Tab. Оболочка добавит lude, дополнив строку до полного имени каталога */usr/include*. Теперь можно нажать Enter, и команда будет выполнена.

Критерием определения имени файла является минимальная полнота. Введите символы в количестве, достаточном, чтобы по ним можно было отличить требуемое имя файла или каталога от всех других в этом каталоге. Оболочка сможет отыскать это имя и дополнить его – вплоть до символа слэша, если найденное имя является именем каталога.

Функция автодополнения может использоваться и с командами. Например, если ввести

```
$ ema
```

и нажать клавишу Tab, оболочка добавит символы cs, дополнив имя команды до emacs (если только какая-либо другая команда в вашем каталоге не начинается с ema).

А что произойдет, если будет найдено несколько файлов, чьи имена соответствуют введенным символам? Если все они начинаются с одних и тех же символов, оболочка допишет слово до того места, где имена файлов начнут различаться. Больше, как правило, оболочки ничего не делают. В *bash* есть изящное усовер-

шенствование: при двойном нажатии клавиши Tab выводятся все возможные варианты дополнения слова. Например, вы вводите:

```
$ cd /usr/l
```

и дважды нажимаете клавишу Tab, тогда *bash* выведет что-нибудь типа:

```
lib         local
```

Командная оболочка *zsh* продвинулась в этом направлении еще дальше: если нажать клавишу Tab еще раз, будет подставлен первый из возможных вариантов; если нажать Tab еще раз, будет подставлен второй вариант, и так далее. Благодаря такой возможности можно не снимать палец с клавиши Tab и добиться нужного результата, не вводя никаких дополнительных символов.

Перемещение по командам

Нажмите клавишу «стрелка вверх», и появится предыдущая введенная вами команда. «Стрелка вверх» перемещает вас по буферу истории команд назад, в то время как «стрелка вниз» перемещает вперед. Если нужно изменить символ в текущей строке, используйте левую или правую стрелку.

Предположим, вы попытались выполнить команду:

```
$ mroe .bashrc
bash: mroe: command not found
```

Очевидно, вместо `more` была введена ошибочная команда `mroe`. Чтобы исправить команду, вызовите ее снова стрелкой вверх. Затем нажимайте левую стрелку, пока курсор не переместится на символ `o` в слове `mroe`. Можно удалить символы `o` и `r` клавишей Backspace и правильно ввести их заново. Но есть более изящная комбинация: нажмите Ctrl+T. В результате `o` и `r` поменяются местами, и можно нажать Enter, чтобы выполнить команду.

Некоторые командные оболочки продвинулись в этом направлении еще дальше: если была введена несуществующая команда, как например `mroe`, командная оболочка исправит опечатку. Разумеется, прежде чем принять исправленный вариант, сначала необходимо убедиться в его правильности, чтобы избежать непреднамеренного исполнения потенциально опасной команды, которая, например, может удалить все ваши файлы.

Для редактирования командной строки есть много других комбинаций клавиш, но даже те, что были продемонстрированы здесь, значительно облегчат вам жизнь. Если вы изучите редактор Emacs, то обнаружите, что большая часть его быстрых комбинаций клавиш работают и в командной оболочке. А если вы любитель *vi*, то можете настроить оболочку так, что она будет использовать привязку клавиш *vi*, а не Emacs. Для этого в *bash*, *ksh* или *zsh* нужно ввести команду:

```
$ export VISUAL=vi
```

В *tcsh* аналогичная команда выглядит следующим образом:

```
$ setenv VISUAL vi
```

Расширение имен файлов

Еще один способ сократить время при вводе – это использование специальных символов для сокращения имен файлов. С помощью этих символов можно задать сразу много файлов. Эта функция оболочки иногда называется «globbing» (универсализация файловых имен).

В интерпретаторе командной строки Windows есть несколько функций такого типа, но они менее развиты. В нем можно использовать знак вопроса, означающий «любой символ», и звездочку, означающую «любую строку символов». В UNIX тоже есть эти символы-заместители, но они используются более ясно и строго.

Допустим, у вас есть каталог, содержащий следующие файлы с исходными текстами на языке C:

```
$ ls
inv1jig.c    inv2jig.c    inv3jig.c    invinitjig.c    invpar.c
```

Чтобы вывести имена трех файлов, содержащие цифры, можно воспользоваться командой:

```
$ ls inv?jig.c
inv1jig.c    inv2jig.c    inv3jig.c
```

Оболочка обнаружит единственный символ, которым можно заменить знак вопроса. Поэтому она выведет список из трех файлов: *inv1jig.c*, *inv2jig.c* и *inv3jig.c*, но не *invinitjig.c*, поскольку в последнем имени слишком много символов.

Если второй файл вас не интересует, можно получить нужный список с помощью квадратных скобок:

```
$ ls inv[13]jig.c
inv1jig.c    inv3jig.c
```

Если после подстановки какого-либо символа из скобок полученный файл соответствует имеющемуся, то имя этого файла выводится. В скобках можно определять диапазон символов:

```
$ ls inv[1-3]jig.c
inv1jig.c    inv2jig.c    inv3jig.c
```

У нас снова получился список из трех файлов. Дефис означает «совпадение с любым символом от 1 до 3 включительно». Можно потребовать совпадения с любой цифрой, указав диапазон `0-9`, и с любой буквой, указав диапазон `[a-zA-Z]`. В последнем случае было задано два диапазона, поскольку оболочка различает верхний и нижний регистры символов. Кстати, использованный порядок основан на наборе символов ASCII.

Предположим, что вы хотите увидеть в списке и файл *init*. Тогда можно воспользоваться звездочкой, поскольку требуется соответствие с любым числом символов между `inv` и `jig`:

```
$ ls inv*jig.c
inv1jig.c    inv2jig.c    inv3jig.c    invinitjig.c
```

Звездочка фактически означает «нуль или больше символов», поэтому, если бы существовал файл с именем *invjig.c*, его имя также было бы выведено на экран.

В отличие от интерпретатора командной строки Windows, командные оболочки UNIX позволяют использовать специальные и обычные символы в любом сочетании. Скажем, вы хотите увидеть все исходные (*.c*) или объектные (*.o*) файлы, содержащие в имени цифру. В следующем шаблоне соединены вместе все шаблоны, изученные нами в этом разделе:

```
$ ls *[0-9]*.[co]
```

Расширения имен файлов очень удобно использовать в сценариях командной оболочки (программах), когда точно не известно, сколько существует файлов. Пусть, например, нужно обработать несколько файлов журналов с именами *log001*, *log002* и т. д. Сколько бы их ни было, все они соответствуют шаблону *log**.

И снова оболочка *zsh* оказалась впереди всех остальных. С помощью *zsh* можно отыскать необходимые файлы не только в текущем каталоге, но и во всех вложенных подкаталогах, для этого достаточно воспользоваться шаблоном **, определяющим имя каталога. Если вернуться к предыдущему примеру поиска определенных файлов с исходными текстами на языке C, но на этот раз попытаться отыскать все файлы во вложенных подкаталогах, то команда будет выглядеть следующим образом:

```
$ ls **/inv?jig.c
inv1jig.c    inv2jig.c    inv3jig.c    old/inv1jig.c
old/veryold/inv1jig.c
```

Расширения имен файлов – это не то же самое, что регулярные выражения, используемые во многих утилитах для задания групп строк. Рассмотрение регулярных выражений выходит за рамки этой книги, но они описаны во многих книгах, рассказывающих об утилитах UNIX. Мы встретимся с регулярными выражениями в главе 19.

Сохранение выводимых данных

По экрану компьютера системного администратора (да и простого человека) проносится масса важных сообщений. Часто желательно сохранить эти сообщения, чтобы позднее изучить или послать другу, который сможет разобраться в том, что же произошло. Поэтому в данном разделе мы немного расскажем о перенаправлении – мощной функции, поддерживаемой оболочками UNIX. Если вы знакомы с Windows, то, вероятно, сталкивались с подобным, но более ограниченным типом перенаправления.

Если после любой команды добавить знак «больше» (>) и вслед за ним указать имя файла, то данные, выводимые этой командой, будут посланы в заданный файл. Например, чтобы перехватить и сохранить результаты работы команды `ls`, можно ввести:

```
$ ls /usr/bin > ~/Binaries
```

Перечень файлов, находящихся в каталоге */usr/bin*, будет записан в файл *Binaries* в вашем домашнем каталоге. Если файл *Binaries* уже существует, то > сотрет его содержимое и заменит результатами работы команды `ls`. Перезапись существующего файла является частой ошибкой пользователей. Если вы работаете в оболочке *csh* или *tcsh*, перезапись можно предотвратить командой

```
$ set noclobber
```

В *bash* того же эффекта можно добиться достичь командой:

```
$ noclobber=1        Не обязательно 1, годится любое ненулевое число
```

Другим и более полезным способом предотвращения перезаписи является добавление новых данных к файлу. Например, сохранив содержимое каталога */usr/bin*, мы хотим добавить к тому же файлу содержимое каталога */bin*. Можно дописать его в конец файла *Binaries*, введя два знака «больше»:

```
$ ls /bin >> ~/Binaries
```

Прием перенаправления вывода оказывается очень удобным, когда нужно неоднократно выполнить одну и ту же утилиту и сохранить результаты для анализа ошибок.

В большинстве программ UNIX есть два потока вывода. Один называется *стандартным выводом* (*standard output*), а другой – *стандартным выводом ошибок* (*standard error*). Если вы программируете на языке C, то сразу их узнаете: стандартный поток вывода ошибок – это дескриптор файла *stderr*, в который выводятся сообщения.

Символ > не перенаправляет поток вывода ошибок. И это удобно, если вам нужно сохранить нормальный вывод, не портя файла с сообщениями об ошибках. Но что если нужно сохранить как раз сообщения об ошибках, как часто бывает при поиске неполадок? Решением является использование знака «больше» с амперсандом (эта конструкция работает почти во всех современных командных оболочках). При этом перенаправляются как поток стандартного вывода, так и поток вывода ошибок. Например:

```
$ gcc invinitjig.c >& error-msg
```

Эта команда сохранит все сообщения компилятора *gcc* в файле с именем *error-msg*. В оболочке Борна и *bash* тот же результат можно получить немного иначе:

```
$ gcc invinitjig.c &> error-msg
```

Теперь представим, что у нас возник такой каприз: сохранить только сообщения об ошибках, а не обычный вывод, то есть поток вывода ошибок, а не поток стандартного вывода. В Борн-совместимых оболочках это можно сделать, дав следующую команду:

```
$ gcc invinitjig.c 2> error-msg
```

Оболочка присваивает номер 1 потоку стандартного вывода и номер 2 потоку вывода ошибок. Поэтому описанная команда сохраняет только поток вывода ошибок.

Наконец, допустим, что вы захотели подавить поток стандартного вывода – не дать выводу появиться на экране. Решение состоит в том, чтобы перенаправить его в особый файл с именем */dev/null*. (Вам не приходилось слышать выражение «Пошлите свою критику в */dev/null!*»? Отсюда и происходит эта фраза.) Каталог */dev* – это место, где UNIX хранит специальные файлы, связанные с терминалами, ленточными приводами и другими устройствами. Но */dev/null* – уникальное устройство: это место, где объекты исчезают, как в черной дыре. Например, следующая команда сохраняет поток вывода ошибок и подавляет поток стандартного вывода:

```
$ gcc invinitjig.c 2>error_msg >/dev/null
```

Теперь вы можете выделить именно тот вывод, который вам нужен.

Если вас заинтересовало, имеет ли знак «меньше» (<) какой-либо смысл для оболочки, то ответ будет утвердительным. Этот знак заставляет команды принимать входные данные из файла. Однако большинство команд и так позволяют задавать входные файлы в командной строке, поэтому в перенаправлении ввода редко возникает необходимость.[1]

Иногда требуется, чтобы одна утилита обрабатывала данные, выводимые другой утилитой. Например, с помощью команды `sort` можно упорядочить результаты работы других команд. Грубый способ добиться этого – сохранить вывод команды в файле и обработать его командой `sort`. Например:

```
$ du > du_output
$ sort -n du_output
```

UNIX предоставляет более простой и эффективный способ выполнения подобных операций с помощью *конвейера* (*pipe*). Нужно лишь поместить вертикальную черту между первой и второй командами:

```
$ du | sort -n
```

Оболочка перешлет весь вывод программы `du` программе `sort`.[2]

В этом примере команда `du` означает «disk usage» («использование диска») и показывает, сколько блоков занимает каждый файл в текущем каталоге. Обычно ее вывод имеет отчасти случайный порядок:

```
$ du
10 ./zoneinfo/Australia
13 ./zoneinfo/US
9 ./zoneinfo/Canada
4 ./zoneinfo/Mexico
5 ./zoneinfo/Brazil
3 ./zoneinfo/Chile
20 ./zoneinfo/SystemV
118 ./zoneinfo
298 ./ghostscript/doc
183 ./ghostscript/examples
3289 ./ghostscript/fonts
.
.
.
```

Поэтому мы решили пропустить результаты ее работы через команду `sort` с параметрами `-n` и `-r`. Параметр `-n` означает «сортировать в числовом (numerical) порядке», а не в порядке ASCII, используемом по умолчанию. Параметр `-r` означает «сортировать в обратном (reverse) порядке», чтобы вначале появлялись самые

---

[1] Хотя, напротив, необходимость получить ввод из заранее подготовленного файла вместо стандартного ввода с клавиатуры, возникает достаточно часто. – *Примеч. науч. ред.*

[2] Точнее, на стандартный канал ввода программы `sort`. – *Примеч. науч. ред.*

большие числа. В результате мы получим вывод, который покажет, какие каталоги и файлы заняли больше всего места:

```
$ du | sort -rn
34368 .
16005 ./emacs
16003 ./emacs/20.4
13326 ./emacs/20.4/lisp
4039 ./ghostscript
3289 ./ghostscript/fonts
.
.
.
```

Поскольку файлов очень много, лучше воспользоваться еще одним конвейером для команды `more` (одно из наиболее частых применений конвейеров):

```
$ du | sort -rn | more
34368 .
16005 ./emacs
16003 ./emacs/20.4
13326 ./emacs/20.4/lisp
4039 ./ghostscript
3289 ./ghostscript/fonts
.
.
.
```

Альтернативой `more` может служить команда `head`, которая выводит лишь несколько первых строк (по умолчанию 10). Конечно, если есть `head`, должна быть и команда `tail`, которая выводит несколько последних строк.

Возможно, вы обратили внимание, что сама по себе команда `du` начинает выводить результаты достаточно быстро и затем, по мере работы, добавляет все новые и новые строки, но когда вывод результатов отправляется через конвейер команде `sort`, информация на экране появляется с задержкой (если жесткий диск будет иметь большой объем и на нем будет храниться достаточно большое число файлов). Это происходит потому, что команда `sort` сначала должна получить полный объем данных, прежде чем выполнить их сортировку, а вовсе не потому, что пересылка по конвейеру сопряжена с большими задержками.

Что такое команда?

Мы говорили, что в UNIX имеется масса разнообразных команд и что можно добавлять новые команды. Это в корне отличает UNIX от большинства операционных систем, содержащих строго ограниченный набор команд. Так что же такое команды UNIX и как они хранятся? В UNIX команда – это просто файл.[1] Напри-

[1] Это относится к большинству, но не ко всем командам UNIX: некоторые простые команды реализуются непосредственно внутри оболочки shell ее средствами, без загрузки внешней программы из файла; примером (не единственным) может служить команда `nice`. – *Примеч. науч. ред.*

мер, команда `ls` является исполняемым файлом, расположенным в каталоге *bin*. Поэтому вместо команды `ls` можно ввести полное имя исполняемого файла, так называемый *абсолютный путь к файлу*:

```
$ /bin/ls
```

Благодаря этому UNIX является очень мощной и гибкой системой. Чтобы предоставить новую утилиту, системный администратор может просто установить ее в стандартный каталог, где размещаются команды. Могут существовать различные версии команды – например, можно предложить для тестирования новую версию утилиты, оставив старую версию в другом месте, а пользователи выберут ту, которую захотят.

При этом часто возникает проблема: вы вводите команду, которая, как вы рассчитываете, есть в системе, но получаете сообщение, подобное «Not found» («Не найдена»). Проблема может быть в том, что команда размещается в таком каталоге, где оболочка ее не ищет. Список каталогов, где оболочка отыскивает команды, называется *«путь поиска»*. Введите следующую команду, чтобы увидеть свой путь поиска (не забудьте про символ доллара, иначе вы увидите не содержимое переменной окружения, а ее имя, которое вам и так известно!):

```
$ echo $PATH
/usr/local/bin:/usr/bin:/usr/X11R6/bin:/bin:/usr/lib/java/bin:\
/usr/games:/usr/bin/TeX:.
```

На этот пример стоит посмотреть внимательнее. Прежде всего, слово `PATH` имеет особый смысл для командной оболочки – оно определяет имя *переменной окружения*. Это краткое имя некоторой полезной информации, в данном случае – это список каталогов, где командная оболочка будет пытаться искать команды. Существует еще несколько переменных окружения, с одной из них, которая называется `SHELL`, мы уже встречались в разделе «Командные оболочки». Когда в командной строке вставляется переменная окружения, перед ее именем должен ставиться символ `$`.

Результатом работы команды `echo` будет последовательность путей, разделенных двоеточиями. Первый путь у данного пользователя – */usr/local/bin*, второй – */usr/bin*, и т. д. Поэтому, если существуют две версии команды, одна в */usr/local/bin*, а другая в */usr/bin*, выполнится та, которая находится в каталоге */usr/local/bin*. Последний путь в этом примере представлен просто точкой, что означает ссылку на текущий каталог. В отличие от командного интерпретатора Windows, UNIX не выполняет поиск в вашем текущем каталоге автоматически. Для этого нужно дать явное указание, как здесь и показано. Некоторые считают, что разрешать поиск в текущем каталоге не очень хорошо по соображениям безопасности. (Злоумышленник, получивший доступ к вашей учетной записи, может скопировать вредоносную программу в один из ваших рабочих каталогов.) Однако это относится в основном к суперпользователю, а обычным пользователям об этом обычно не нужно беспокоиться.

Если команда не найдена, нужно установить, где она находится, и добавить соответствующий каталог к пути поиска. О местонахождении команды должно быть сказано на страницах руководства. Допустим, вы обнаружили ее в */usr/sbin*, где находится ряд команд системного администрирования. Вам ясно, что требуется доступ к этим командам системного администрирования, поэтому вве-

дите следующую команду (обратите внимание, перед первым словом PATH нет символа доллара, а перед вторым есть):

```
$ export PATH=$PATH:/usr/sbin
```

Эта команда добавит каталог */usr/sbin*, но сделает его последним каталогом в списке. Эта команда означает: «Сделать мой путь равным прежнему пути плюс */usr/sbin*».

Такая команда работает не во всех оболочках. Для большинства пользователей Linux, работающих в Борн-совместимых оболочках типа *bash*, она действует прекрасно, но в *csh* или *tcsh* вместо нее нужно ввести следующую команду:

```
set path = ( $PATH /usr/sbin )
```

Наконец, есть несколько команд, которые не являются исполняемыми файлами, и одна из них – команда cd. В большинстве своем эти команды оказывают воздействие на саму оболочку и потому должны интерпретироваться и выполняться самой оболочкой. Поскольку они являются частью оболочки, то называются *встроенными командами*.

Запуск команды в фоновом режиме

Независимо от того, работаете ли вы в X Window System (будет описана ниже) или в виртуальных консолях, существует возможность одновременного запуска нескольких команд из одной и той же командной оболочки и избежать необходимости постоянно переключаться между окнами или консолями. Для этого достаточно воспользоваться преимуществами многозадачности UNIX, помещая в конец команды символ амперсанда, как показано в следующем примере:

```
$ gcc invinitjig.c &
[1] 21457
```

Амперсанд запускает команду в фоновом режиме, который означает, что приглашение оболочки возвращается и вы можете снова выполнять другие команды, пока *gcc* компилирует вашу программу. Число [1] – это номер задания, присвоенный вашей команде. Число 21457 – это идентификатор процесса, о котором мы поговорим позже. Номера заданий присваиваются командам, запущенным в фоновом режиме, по порядку, и потому их легче запоминать и вводить, чем идентификаторы процессов.

Разумеется, многозадачность достигается не бесплатно. Чем больше команд запущено в фоновом режиме, тем медленнее работает система, поскольку чередует их выполнение.

Не следует запускать команду в фоновом режиме, если она должна получать данные, вводимые пользователем. Если сделать это, появится сообщение об ошибке:

```
Stopped (tty input)
```

Решить эту проблему можно, переведя задачу обратно на передний план командой *fg*. Если в фоновом режиме выполняется много команд, то выбрать одну из них можно, указав номер задания или идентификатор процесса. Для нашей долгоиграющей команды gcc эквивалентны следующие команды:

```
$ fg %1
```

```
$ fg 21457
```

Не забудьте поставить знак процента перед номером задания: он позволяет отличать номера заданий от идентификаторов процессов.

Прекратить выполнение команды в фоновом режиме можно командой *kill*:

```
$ kill %1
```

Чтобы перевести в фоновый режим уже запущенную команду, в большинстве командных оболочек достаточно нажать комбинацию клавиш Ctrl+Z. Она временно приостановит исполнение текущей программы переднего плана. После этого можно дать команду *fg*, которая уже была описана выше, чтобы возобновить работу программы на переднем плане, или команду *bg*, которая переведет исполнение программы в фоновый режим.

Регистрация и исполнение команд в удаленной системе

Вероятно, ваш компьютер подключен к офисной или домашней локальной сети либо имеет коммутируемое подключение к Интернету. Иногда возникает необходимость войти в другую систему или скопировать файл в или из другой системы.

Если вам необходима помощь в настройке сети, обратитесь к главе 13 и следующим за ней главам. В этом разделе мы будем исходить из предположения, что сеть уже подключена и настроена. Если вы в состоянии просмотреть веб-страницу в Интернете с помощью веб-броузера, следовательно, сеть подключена и вы можете опробовать примеры из этого раздела. В своих примерах мы будем использовать пакет по имени SSH, который входит в состав большинства, если не всех, дистрибутивов Linux.

Аббревиатура «SSH» происходит от «Secure Shell» (безопасная командная оболочка). Название говорит о том, что разработчики сосредоточили свои усилия на достижении максимальной защищенности сетевого соединения от злоумышленников. SSH стал чрезвычайно уважаемым и популярным протоколом взаимодействия с удаленными системами и поддерживается самыми разными системами, такими как графический интерфейс Putty для Windows (*http://www.chiark.greenend.org.uk/~sgtatham/putty*).

В Linux используется свободно распространяемая реализация SSH – OpenSSH (*http://www.openssh.com*). Ошибки в ней встречаются достаточно редко, хотя время от времени ошибки все-таки обнаруживаются, поэтому из соображений безопасности желательно регулярно обновлять свой дистрибутив. OpenSSH поддерживает последний стандарт – протокол SSH версии 2. Если вам понадобится реализовать какое-нибудь сложное сетевое решение на основе SSH, то необходимо будет гораздо глубже изучить SSH. В этом вам поможет книга «SSH, The Secure Shell: The Definitive Guide» (O'Reilly).

В этом разделе будут представлены четыре-пять команд, которые вы будете использовать чаще всего. Предположим, что у вас есть учетная запись с именем *mdw* на удаленной системе, которая называется *eggplant*. Тогда зарегистрироваться в удаленной системе можно будет следующим образом:

```
$ ssh -l mdw eggplant
```

Параметр `-l` задает имя пользователя в удаленной системе. Другой вариант регистрации, идентичный предыдущему:

```
$ ssh mdw@eggplant
```

Если имя учетной записи в локальной и удаленной системах совпадает, то имя пользователя можно не указывать:

```
$ ssh eggplant
```

Каждый раз, когда запускается сеанс `ssh`, у вас будет запрошен пароль учетной записи в удаленной системе.

Если в ходе сеанса возникнет необходимость сделать что-нибудь на локальной машине, вам не нужно будет закрывать его или переключаться в другое окно. Просто приостановите сеанс, введя символ тильды (`~`), и затем нажмите комбинацию Ctrl+Z. (Иногда символ тильды не опознается протоколом SSH с первой попытки, если это случится, повторите попытку. Если символ тильды не появился на экране, значит, все в порядке.) Возобновить работу приостановленного сеанса можно с помощью команды *fg* точно так же, как и работу любой другой приостановленной программы.

Если вам не нужен доступ к удаленной командной оболочке, а нужно всего лишь запустить единственную команду на удаленной системе, просто введите нужную команду после сетевого имени системы:

```
$ ssh -l mdw eggplant rm logfiles/temp_junk
```

Или, если имена учетной записи в локальной и удаленной системах совпадают:

```
$ ssh eggplant rm logfiles/temp_junk
```

Имена файла, как *logfiles/temp_junk* в данном примере, интерпретируются так, как будто вы находитесь в своем домашнем каталоге (каталог, который назначается текущим сразу после регистрации в системе). Если вам нужна полная уверенность, что действие будет выполнено над нужным файлом в нужном каталоге, используйте абсолютные имена (например, */home/mdw/logfiles/temp_junk*).

Страницы справочного руководства по команде *ssh* содержат массу интереснейшей информации: из них вы узнаете, как запускать приложения X Window System с графическим интерфейсом поверх SSH и как избежать раздражающей необходимости вводить пароль перед вызовом каждой команды. (Хотя в этом случае вам придется «поплясать» вокруг файлов с настройками, о которых ничего не говорится в справочном руководстве.)

Чтобы скопировать файл, следует воспользоваться другой командой из пакета SSH – `scp`. Ниже приводится пример копирования файла из локальной системы в удаленную:

```
$ scp logfiles/temp_junk mdw@eggplant:
```

Здесь также имя пользователя и символ `@` могут быть опущены, если имена учетной записи в локальной и удаленной системах совпадают. (Обратите внимание: в данном случае синтаксис с параметром *-l* не использовался, поскольку в команде *scp* параметр *-l* имеет совсем другое назначение.)

Не забудьте завершить команду символом двоеточия; без него файл просто будет скопирован в другой файл с именем *eggplant* в локальной системе. Данная команда скопирует файл на удаленную систему *eggplant*, в ваш домашний каталог (как и в случае с *ssh*). Вы можете скопировать файл в любой другой каталог, указав путь к нему относительно домашнего каталога или определив абсолютный путь.

Чтобы выполнить обратную операцию – скопировать файл из удаленной системы в локальную, можно ввести такую команду:

```
$ scp mdw@eggplant:logfiles/temp_junk .
```

Точка в конце команды означает текущий каталог в локальной системе. Вместо нее можно указать любой другой относительный или абсолютный путь к требуемому каталогу.

Чтобы скопировать каталог со всем его содержимым, следует использовать параметр `-r`:

```
$ scp -r mdw@eggplant:logfiles .
```

Страницы справочного руководства

У вас появится больше возможностей при наличии сведений о том, как выполнять собственные исследования. Следуя этой заповеди, мы расскажем вам сейчас об интерактивной системе помощи, которая встраивается в UNIX-системы. Она называется *страницами справочного руководства* (*manpages*).

В действительности страницы справочного руководства не являются верхом совершенства, поскольку они кратки и предполагают наличие существенной подготовки по UNIX. Каждая страница посвящена отдельной команде и редко способствует получению представления о том, зачем ее нужно использовать. Тем не менее эти страницы очень важны. В каждой UNIX-системе команды могут иметь некоторые особенности, и страницы руководства – наиболее надежный способ выяснить, что делает именно ваша система. (Участники проекта документирования Linux заслуживают большой благодарности за то, что потратили огромное количество времени на создание страниц справочного руководства.) Справку о команде можно получить, введя, например, следующее:

```
$ man ls
```

Страницы справочного руководства состоят из нескольких разделов, в зависимости от их назначения. Команды пользователя находятся в разделе 1, системные вызовы UNIX – в разделе 2, и т. д. Наибольший интерес для вас будут представлять разделы 1, 5 (форматы файлов) и 8 (команды системного администрирования). Номера разделов очень важны при интерактивном просмотре страниц, так как их можно указывать при поиске команды:

```
$ man 1 ls
```

При чтении печатной копии руководства вы обнаружите, что оно фактически разделено на части согласно принятой схеме нумерации. Иногда в различных разделах встречаются статьи с одинаковыми названиями. (Например, *chmod* является и командой, и системным вызовом.) Поэтому иногда можно увидеть название страницы, за которым в скобках следует номер раздела, например *ls*(1).

Существуют ситуации, в которых номер раздела необходимо вводить в командной строке, а именно: если для одного ключевого слова есть несколько страниц, например для команды с некоторым именем и системной функции с тем же именем. Предположим, что вы хотите посмотреть библиотечный вызов, а система показывает вам описание команды, поскольку по умолчанию сначала ищутся команды. Чтобы увидеть страницу руководства по системному вызову, нужно указать номер раздела, в котором она находится.

Взгляните на верхнюю часть страницы руководства. Первый заголовок – NAME (Название). Под ним расположено краткое описание темы в одну строку. Эти описания полезны, если вы не вполне уверены в том, что разыскиваете. Выберите какое-нибудь слово, связанное с тем, что вы ищете, и укажите его в команде apropos:

```
$ apropos edit
```

Эта команда покажет все страницы справочного руководства, так или иначе связанные с редактированием. Алгоритм очень простой: apropos просто выводит все строки названий, содержащие запрошенную строку.

Многие другие утилиты, в особенности предлагаемые графическими средами, обсуждаемыми в главе 3, выводят страницы руководства в более привлекательном виде.

Как и команды, страницы руководства иногда располагаются в самых неожиданных местах. Например, некоторые специфические программы могут быть установлены в каталог */usr/local*, а их страницы справочного руководства помещены в */usr/local/man*. Команда man не станет автоматически осуществлять поиск в */usr/local/man*, поэтому при запросе страницы руководства можно получить сообщение «No manual entry». Чтобы исправить положение, нужно задать все каталоги верхнего уровня в переменной окружения MANPATH. Например, вам нужно вставить фактические каталоги, в которых размещены страницы руководств в вашей системе:

```
$ export MANPATH=/usr/man:/usr/local/man
```

Синтаксис такой же, как в случае с переменной PATH, описанной выше: каталоги разделены двоеточиями. Если вы работаете в оболочке *csh* или *tcsh*, нужно ввести:

```
$ setenv MANPATH /usr/man:/usr/local/man
```

Есть еще одна переменная окружения, которую, возможно, придется установить, – MANSECT. Она определяет порядок просмотра разделов при поиске статей руководства. Например, следующая строка

```
$ export MANSECT=«2:3:1:5:4:6:7:8:n:9»
```

указывает, что поиск должен начинаться с раздела 2.

Вы прочли несколько страниц руководства и все еще пребываете в недоумении? Они не предназначены для введения в незнакомые темы. Достаньте хорошую книгу по UNIX для начинающих и возвращайтесь к страницам руководства постепенно, по мере ознакомления с системой. Со временем они станут для вас незаменимыми.

Страницы справочного руководства – не единственный источник информации в UNIX-системах. У программ проекта GNU часто есть информационные страни-

цы Info, которые читаются с помощью программы *info*. Например, чтобы прочесть страницы Info для команды `find`, нужно ввести:

```
info find
```

Программа *info* загадочна и имеет массу функций для навигации. Чтобы разобраться в ней, лучше всего, находясь в программе, нажать Ctrl+H и прочесть весь экран помощи. К счастью, есть программы, которые упрощают чтение страниц Info, например *tkinfo* и *kdehelp*. Эти команды используют X Window System для реализации графического интерфейса. Можно читать страницы Info и в Emacs (см. раздел «Учебное руководство и оперативная подсказка» в главе 19) либо использовать команду `pinfo`, имеющуюся в некоторых дистрибутивах Linux и работающую сходным с веб-броузером Lynx образом.

В последнее время документация все чаще поставляется в виде страниц HTML. Их можно читать любым веб-броузером (глава 5). Например, в веб-броузере Konqueror вы выбираете Open Location (Открыть страницу) в меню Location (Адрес) и нажимаете кнопку с изображением папки, в результате чего открывается обычное окно диалога, в котором можно выбрать нужный файл документации. Иногда документация может распространяться в виде файлов PDF, которые можно просматривать либо с помощью коммерческой программы Acrobat Reader, которая входит в состав большинства дистрибутивов и которую можно загрузить с сайта *http://www.adobe.com*, либо с помощью свободно распространяемых программ *xpdf* и *KGhostview*.

Стартовые файлы

Возможность настройки является сильной стороной UNIX. Корни этого, возможно, лежат в двух характерных для хакеров чертах: стремление к полному контролю над своей средой и минимизация нажатий клавиш и прочих необходимых телодвижений. Поэтому все основные утилиты в UNIX – редакторы, почтовые программы, отладчики, клиенты X Window System – имеют файлы, с помощью которых можно в значительной степени переопределить поведение по умолчанию. Имена этих файлов часто оканчиваются на *rc* (от «resource configuration» – конфигурации ресурсов).

Стартовые файлы обычно находятся в домашнем каталоге пользователя. Их имена начинаются с точки, благодаря чему команда `ls` обычно их не выводит. Наличие стартовых файлов не является необходимым: все связанные с ними программы используют значения по умолчанию, если файл не существует. Но иметь стартовые файлы всегда удобно. Ниже описаны некоторые из наиболее распространенных стартовых файлов:

.bashrc

Используется командной оболочкой *bash*. Файл является сценарием командной оболочки, то есть содержит команды и другие программные конструкции. Ниже приводится пример короткого стартового файла, который мог быть помещен в домашний каталог программой, создавшей учетную запись:

```
PS1='\u$'    # В приглашение к вводу включается имя пользователя.
HISTSIZE=50 # Сохранять последние 50 команд для вызова нажатием стрелки вверх.
# Список каталогов, в которых ведется поиск команд.
```

```
PATH=/usr/local/bin:/usr/bin:/bin:/usr/bin/X11
# Чтобы не допустить случайного окончания сеанса регистрации,
# запретить выход при нажатии комбинации Ctrl+D.
IGNOREEOF=1
stty erase "^H" # Обеспечить стирание при нажатии backspace.
```

.bash_profile

Используется командной оболочкой *bash*. Это тоже сценарий на языке командной оболочки. Различие между этим сценарием и *.bashrc* в том, что *.bash_profile* выполняется только при входе в систему. Он был разработан для того, чтобы можно было отделить интерактивные оболочки от тех, которые выполняются фоновыми обработчиками типа *cron* (описывается в главе 10). Но на современных компьютерах с X Window System он почти не используется, поскольку при открытии окна терминала выполняется только *.bashrc*. Если вы открываете окно командой `xterm -ls`, то *.bash_profile* тоже выполняется.

.zhrc

Аналог *.bashrc* для командной оболочки *zsh*

.zprofile

Аналог *.bash_profile* для командной оболочки *zsh*

.cshrc

Используется командной оболочкой C или *tcsh*. Файл является сценарием, использующим языковые конструкции оболочки C.

.login

Используется командной оболочкой C или *tcsh*. Файл является сценарием, использующим языковые конструкции оболочки C. Как и *.bash_profile* в оболочке *bash*, он выполняется только при входе в систему. Ниже приводится пример содержимого файла *.cshrc* или *.login*:

```
set prompt='$ ' # Простое приглашение к вводу в виде $.
set history=50  # Хранить 50 команд для вызова нажатием стрелки вверх.
# Список каталогов, в которых ведется поиск команд.
set path=(/usr/local/bin /usr/bin /bin /usr/bin/X11)
# Чтобы не допустить случайного окончания сеанса регистрации,
# запретить выход при нажатии комбинации Ctrl+D.
set ignoreeof
# Обеспечить стирание при нажатии backspace.
stty erase "^H"
```

.emacs

Используется редактором Emacs. Состоит из функций LISP. См. раздел «Настройка Emacs» в главе 19.

.exrc

Используется редактором *vi* (визуальный редактор, совмещенный с более старым редактором *ex*). Каждая строка является командой редактора. См. раздел «Расширение возможностей vi» в главе 19.

.newsrc

Используется программами чтения телеконференций. Содержит список всех телеконференций, имеющихся на сайте.

.xinitrc

Используется X Window System. Представляет собой сценарий на языке командной оболочки, который запускается в начале сеанса X. В разделе «Запуск X» главы 16 подробно описывается порядок использования этого файла

.kde/share/config

Фактически это целый каталог с файлами настроек графической среды K Desktop Environment (KDE). Там находится множество файлов, имена которых начинаются с имени программы, настройки которой они содержат, и оканчиваются *rc*. Эти файлы обычно не требуется редактировать вручную: почти все программы поставляются со своими утилитами настройки. В зависимости от версии KDE этот путь может начинаться с *.kde2* или *.kde3*.

.gnome

Как и в предыдущем случае, это целый каталог файлов конфигурации, на этот раз для графической среды GNOME.

Важные каталоги

Вам уже известен каталог */home*, в котором хранятся файлы пользователей. Для вас как для системного администратора и программиста будут важны и некоторые другие каталоги. Ниже приводятся некоторые из них с кратким описанием их содержимого:

/bin

Основные команды UNIX, такие как *ls*.

/usr/bin

Другие команды. Различие между */bin* и */usr/bin* весьма условное; в старых системах UNIX с маленькими дисками было удобно разделить команды.

/sbin

Команды общего назначения, используемые суперпользователем для системного администрирования.

/usr/sbin

Команды, используемые суперпользователем для системного администрирования.

/boot

Место, где иногда хранятся ядро и другие файлы, используемые при загрузке.

/etc

Файлы, используемые такими подсистемами, как сетевая подсистема, NFS и электронная почта. Обычно в них содержатся таблицы сетевых служб, монтируемых дисков и тому подобное. Многие файлы из этого каталога используются для загрузки системы или отдельных служб, и мы еще встретимся с ними.

/var

Административные файлы, такие как журналы, используемые различными утилитами.

/var/spool

Временное хранилище для файлов, передаваемых на печать, пересылаемых по UUCP и тому подобное.

/usr/lib

Стандартные библиотеки, например *libc.a*. При сборке программы компоновщик ищет здесь файлы, указанные в параметрах *-l*.

/usr/lib/X11

Дистрибутив X Window System. Содержит библиотеки, используемые клиентами X, а также шрифты, образцы файлов ресурсов и другие важные части пакета X. Этот каталог обычно является символической ссылкой на */usr/X11R6/lib/X11*.

/usr/include

Стандартное местоположение заголовочных файлов программ на языке C, таких как *<stdio.h>*.

/usr/src

Здесь находятся исходные тексты программ, собираемых в системе.

/usr/local

Программы и файлы данных, локально добавленные системным администратором.

/etc/skel

Примеры стартовых файлов, которые помещаются в домашние каталоги новых пользователей.

/dev

Этот каталог содержит так называемые *файлы устройств* – интерфейс между файловой системой и аппаратурой (например, */dev/modem* представляет в системе модем).

/proc

Подобно тому как */dev* представляет интерфейс между файловой системой и аппаратурой, */proc* представляет интерфейс между файловой системой и выполняющимися процессами, ЦПУ и памятью. Содержащиеся здесь файлы (это не настоящие, а виртуальные файлы, генерируемые динамически, когда нужно их просмотреть) могут предоставить информацию об окружении определенного процесса, состоянии и конфигурации ЦПУ, настройке портов ввода-вывода и т. д.

/opt

Каталог */opt* часто используется для хранения крупных пакетов программного обеспечения. Например, весьма вероятно, что окружение рабочего стола KDE будет установлено в каталог */opt/kde3* (или */opt/kde4*, как только появится версия с номером 4), пакет офисных программ OpenOffice – в каталог */opt/OpenOffice.org*, а веб-броузер Firefox – в каталог */opt/firefox*.

Основы редактирования текста

Теперь, когда мы ознакомились с некоторыми файлами настроек, необходимо рассмотреть основы их редактирования. Основное обсуждение различных текстовых редакторов будет идти в главе 19.

В качестве примера мы будем использовать здесь текстовый редактор Emacs, который отличается дружелюбностью и получил широкое распространение. Остальные редакторы, такие как *vi*, получили, может быть, не менее широкую известность, но они не так дружелюбны по отношению к начинающему пользователю. Существуют и не менее дружелюбные редакторы, но они не так известны и могут присутствовать не во всех дистрибутивах Linux. Подробнее о *vi* и других редакторах мы поговорим позднее.

Редактор Emacs поставляется в двух основных версиях: GNU Emacs и XEmacs. Редактор GNU Emacs запускается командой:

```
$ emacs filename
```

а XEmacs – командой:

```
$ xemacs filename
```

Если запуск производится не из графического окружения, необходимо добавить параметр *-nw* (от «no windows» – без окон):

```
$ xemacs -nw filename
```

Скорее всего, в вашей системе присутствует одна из упомянутых версий редактора, и для упрощения изложения материала пока будем считать различия между ними несущественными. Если в системе имеются обе версии, рекомендуем использовать XEmacs.

К настоящему моменту вам достаточно будет знать очень немногое: как редактировать текст, как сохранить результаты своего труда и как завершить работу с редактором. Разумеется, Emacs обладает намного более широкими возможностями, но мы оставим эти темы для обсуждения в будущем.

После запуска Emacs вы увидите содержимое файла, имя которого было задано в командной строке. Теперь можно приступать к редактированию: вводить новый текст, удалять существующий с помощью клавиши Backspace и перемещаться по тексту с помощью клавиш управления курсором. Чтобы сохранить результат своего труда, нужно нажать комбинации клавиш C-x C-s. На жаргоне Emacs это означает: «нажать клавишу Ctrl и, удерживая ее, нажать клавишу X, отпустить обе клавиши, затем опять нажать клавишу Ctrl и, удерживая ее, нажать клавишу S, отпустить обе клавиши». Такой порядок действий первое время может показаться довольно странным, но, выполнив его пару раз, ваши пальцы сами запомнят необходимые комбинации, и вам не придется даже задумываться о них. В некоторых дистрибутивах, но далеко не во всех, Emacs сопровождается даже графическими меню, подобными тем, что используются в других операционных системах, поэтому мы будем придерживаться порядка действий, который гарантированно обеспечивается в любом дистрибутиве.

После того как файл будет сохранен, можно закрыть Emacs. Делается это с помощью комбинаций C-x C-c. Нетрудно догадаться, что это означает буквально следующее: «нажать клавишу Ctrl и, удерживая ее, нажать клавишу X, отпустить

обе клавиши, затем опять нажать клавишу Ctrl и, удерживая ее, нажать клавишу C, отпустить обе клавиши». После этого Emacs закроется, и вы вернетесь обратно в командную строку.

Дополнительные сведения о командных оболочках и сценариях

В этом разделе будут рассмотрены более сложные приемы работы с командной оболочкой – интерпретатором командной строки Linux.

Настройка параметров терминала

Управление различными характеристиками терминала (например, виртуальной консоли), такими как скорость автоповтора нажатой клавиши, величина табулостопов и цвет текста, производится с помощью команды `setterm`.

Большинством пользователей эта команда используется для изменения цвета текста в виртуальных консолях. Это дает возможность сразу определить номер консоли по цвету шрифта. (Обратите внимание: это относится только к виртуальным текстовым консолям, работающим в текстовом режиме. Окна эмуляторов терминала в X11 настраиваются несколько иным образом.)

Например, чтобы изменить цветовые настройки текущего терминала и назначить ему палитру с белым шрифтом на синем фоне, достаточно дать следующую команду:

```
$ setterm -foreground white -background blue
```

Некоторые программы и команды вызывают сброс настроек терминала к значениям по умолчанию. Чтобы сохранить текущие настройки, сделав их настройками по умолчанию, нужно дать команду:

```
$ setterm -store
```

У команды *setterm* имеется масса разнообразных параметров (большинство из которых вы наверняка никогда не будете использовать). Чтобы получить дополнительные сведения об этой команде, обращайтесь к страницам справочного руководства *setterm*(1) или дайте команду *setterm -help*.

Если по какой-либо причине настройки терминала собьются (такое может произойти, например, при попытке просмотреть содержимое двоичного файла с помощью команды *cat*), попробуйте вслепую набрать команду *setterm -reset*, она вернет настройки терминала в обычное состояние

Программирование на языке командной оболочки

В разделе «Командные оболочки» были представлены различные командные оболочки, имеющиеся в Linux, но командные оболочки в умелых руках могут быть мощным и гибким инструментом программирования.[1] Различия между ни-

[1] Достаточно много команд-файлов самой системы, о которых рассказывалось ранее, являются не программами, а сценариями на языках *shell* или Perl. – *Примеч. науч. ред.*

ми становятся более очевидными, если начать писать сценарии на языке командной оболочки. Оболочка Борна и оболочка C имеют определенные отличия в командном языке, которые едва ли удастся обнаружить при работе в обычном, интерактивном режиме. Язык командной оболочки Z являет собой надмножество языка оболочки Борна. Большинство различий проявляются лишь при попытке использовать малоизвестные, специфичные особенности командной оболочки, такие как подстановка слов или расширение параметров функций.

Наиболее заметные отличия между оболочками Борна и C наблюдаются в конструкциях управления потоком исполнения, включая условный оператор `if ... then` и цикл `while`. В оболочке Борна условный оператор `if ... then` имеет следующую форму записи:

```
if list
then
  commands
elif list
then
  commands
else
  commands
fi
```

где *list* – это обычная последовательность команд, используемая как выражение проверки условия в операторах `if` и `elif` (сокращенно от «else if»). Условие считается выполненным, если выражение *list* вернуло код возврата 0 (в отличие от булевых выражений в языке C, командная оболочка воспринимает значение 0 как признак успешного завершения команды). Блоки *commands* – это обычные группы команд, исполняемые в случае, когда выражение *list* возвращает значение 0. Зарезервированное слово `then` после каждой последовательности *list* должно располагаться на следующей строке, чтобы интерпретатор мог отличить его от самой последовательности команд *list*. Как вариант, *then* можно расположить на одной строке с *list*, правда при этом последовательность *list* должна завершаться символом точки с запятой (`;`). То же самое относится и к блокам команд *commands*.

Например:

```
if [ "$PS1" ]; then
  PS1="\h:\w% "
fi
```

Данная последовательность проверяет, является ли текущий экземпляр командной оболочки оболочкой входа (точнее, была ли установлена переменная окружения `PS1`, содержащая шаблон приглашения к вводу). Если переменная была установлена, в нее перезаписывается шаблон `\h:\w%`, при отображении на экране разворачивающийся в имя сетевого узла, за которым следует имя текущего каталога, например:

```
loomer:/home/loomer/mdw%
```

Оператор `[...]`, стоящий вслед за резервированным словом `if` и выполняющий проверку условия, является встроенной командой *bash*. Это краткая форма записи команды *test*. Команда *test* и ее сокращенный эквивалент обеспечивают удобный механизм проверки значений переменных командной оболочки, сравнения строк и т. д. Вместо оператора `[...]` после `if` можно использовать любой

другой набор команд, при этом код завершения последней команды будет определять значение всего условного выражения.

В командной оболочке *tcsh* условный оператор `if ... then` выглядит следующим образом:

```
if (expression) then
  commands
else if (expression) then
  commands
else
  commands
endif
```

Отличие здесь заключается в том, что выражение *expression* после `if` является арифметическим или логическим выражением, которое вычисляется самой оболочкой *tcsh*, тогда как в *bash* условное выражение является командой, а решение о выполнении условия принимается на основе кода завершения, возвращаемого командой. В *bash* команда *test* и оператор `[ ... ]` по своему смыслу совпадают с проверкой арифметического выражения в *tcsh*.

Тем не менее, несмотря на такие отличия в *tcsh* тоже есть возможность запускать внешние команды внутри условного выражения *expression*, для чего нужно лишь окружить выражение фигурными скобками: *{command}*.

Проверку переменной `PS1`, эквивалентную предыдущему примеру, в *tcsh* можно выполнить следующим образом:

```
if ($?prompt) then
  set prompt="%m:%/%% "
endif
```

где для описания шаблона приглашения используются специальные символы, характерные для *tcsh*. Как видно из последнего примера, язык командной оболочки *tcsh* по своему синтаксису напоминает язык программирования C, и выражения в нем являются либо арифметическими, либо логическими. В оболочке *bash*, напротив, практически все выражения являются командами и значения выражений вычисляются в терминах кодов завершения. Между любыми командными оболочками можно найти сходства, но подходы к решению конкретных задач будут несколько отличаться.

Аналогичные различия существуют в синтаксисе цикла с предусловием `while`. В *bash* он записывается следующим образом:

```
while list
do
  commands
done
```

Условие входа в цикл можно изменить на противоположное, для чего вместо зарезервированного слова `while` достаточно подставить зарезервированное слово `until`. Как и прежде, *list* – это всего лишь последовательность исполняемых команд, в которой код завершения последней команды определяет результат всего условного выражения (нуль трактуется как истина, ненулевое значение – как ложь). В *tcsh* цикл с предусловием выглядит следующим образом:

```
while (expression)
  commands
end
```

где *expression* – это логическое выражение, вычисляемое оболочкой *tcsh*.

Этих примеров должно быть достаточно для понимания различий языка командной оболочки *bash* и *tcsh*. Было бы желательно, чтобы вы просмотрели страницы справочного руководства bash(1) и tcsh(1) (хотя основная их цель заключается в том, чтобы дать справочную информацию, а не выступать в роли учебника) и страницы Info, если они у вас установлены. Неплохо будет прочитать какие-нибудь книги или учебники по этим двум оболочкам; фактически подойдет любая книга по программированию на языке командной оболочки. После этого вы сможете с помощью страниц справочного руководства самостоятельно разобраться с особенностями *bash* и *tcsh*, расширяющими возможности стандартных оболочек Борна и C. От себя можем порекомендовать замечательные книги «Learning the bash Shell» Камерона Ньюхама (Cameron Newham) и Билла Розенблатта (Bill Rosenblatt) и «Using csh and tcsh» Поля Дюбуа (Paul DuBois) (обе изданы O'Reilly).

Преимущества командной оболочки Z

Оболочка Z (*zsh*) высоко ценится за многие свои особенности, которые увеличивают эффективность командной строки. Для начала рассмотрим приглашение к вводу. В *zsh* приглашение к вводу делится на две половины – левую и правую. Левая половина определяется содержимым переменной окружения `PROMPT`, а правая – переменной `RPROMPT`, например последовательность команд:

```
export PROMPT="%n@%m"
export RPROMPT="%~%"
```

даст имя пользователя и имя сетевого узла слева от строки ввода и имя текущего каталога – справа. Правая половина приглашения обладает замечательным свойством – она автоматически исчезает, когда нужно освободить место в строке для ввода.

Оболочка *zsh* обладает массой интересных особенностей, например многие ее параметры могут быть изменены с помощью команды `setopt`. Полный список параметров можно найти на страницах справочного руководства *zshoptions*, однако нам хотелось бы упомянуть здесь об одном очень примечательном параметре – `ALL_EXPORT`. После того как будет выполнена команда

```
setopt ALL_EXPORT
```

все переменные окружения, созданные вами, будут экспортироваться автоматически. Это очень удобно для тех, кто, как и мы, подготавливая переменные окружения для программ, отличных от командной оболочки, иногда забывает экспортировать их, а затем удивляется, почему они не воспринимаются процессами, запущенными из командной оболочки. Отключить эту возможность можно командой `setopt noALL_EXPORT`.

Выше уже говорилось о команде `cd`. Разумеется, эта команда также существует и в *zsh*, но при этом она обладает некоторыми интересными особенностями. Например, если ей в качестве аргумента передать символ дефиса (-), она выполнит

переход в каталог, где вы находились перед выполнением последней команды `cd` (в следующем примере мы вернули отображение имени текущего каталога обратно в левую половину приглашения):

```
      ~%> cd kdesvn/kdelibs/kdecore
      ~/kdesvn/kdelibs/kdecore> pwd
/home/kalle/kdesvn/kdelibs/kdecore
      ~/kdesvn/kdelibs/kdecore> cd /usr/local
      /usr/local> cd -
~/kdesvn/kdelibs/kdecore
~/kdesvn/kdelibs/kdecore>
```

Кроме того, если попытаться исполнить неизвестную команду (то есть когда команда не является ни именем исполняемого файла, находящегося в одном из каталогов, содержащихся в пути поиска `PATH`, ни встроенной командой), но при этом имя команды будет совпадать с именем каталога, *zsh* интерпретирует ее как команду перехода в этот каталог:

```
~> Documents
~/Documents>
```

Еще одна интересная особенность – функция автоматического исправления опечаток в именах команд. Например, если вы постоянно пытаетесь ввести команду `mroe` вместо `more`, включите функцию автокоррекции командой:

```
setopt CORRECT
```

После этого *zsh* будет предлагать варианты исправления ошибок, если не сможет распознать вводимые команды:

```
      ~/Documents> mroe /etc/motd
      zsh: correct 'mroe' to 'more' [nyae]? y
Welcome to tigger...
```

Даже такая функция, как автодополнение, в *zsh* наделена массой особенностей, отсутствующих в других командных оболочках. Она способна автоматически дополнить практически все, что угодно. Вы уже знаете, что при нажатии клавиши Tab в процессе ввода команды или имени файла большинство командных оболочек попробуют завершить начатую строку. Но в *zsh* эта функция может гораздо большее:

```
rpm --erase <TAB> # выведет список установленных пакетов

rpm -q<TAB>     # выведет возможные дополнительные параметры для параметра 'q'

fg % <TAB>      # выведет список имен фоновых процессов, которые могут быть
                # перемещены на передний план

cvs checkout <TAB> # выведет список модулей, доступных для выемки из репозитория

make -f Makefile <TAB> # выведет список целей, имеющихся в Makefile

cd <TAB> # выведет только каталоги
```

Функция автодополнения в *zsh* обладает еще многими и многими особенностями, а кроме того, есть возможность добавить в нее свои расширения. Как это сделать – описывается на страницах справочного руководства *zshcompctl*.

5

Веб-броузеры и обмен мгновенными сообщениями

Одним из средств каждодневного общения, полюбившимся миллионам пользователей, являются веб-броузеры и программы обмена мгновенными сообщениями, включая IRC. В состав Linux входят свободные реализации этих инструментов, которые по своим характеристикам порой превосходят коммерческие аналоги.

World Wide Web

Несомненно, каждый, кто имеет хоть какое-то отношение к компьютерам, пользовался в какой-то мере World Wide Web. Как когда-то текстовые процессоры и электронные таблицы заставляли людей обращаться к компьютерам, так сегодня это делает WWW. Мы расскажем о некоторых средствах, с помощью которых можно получить доступ к Web из Linux.

С самого своего возникновения операционная система Linux была тесно связана с Интернетом в целом и с WWW в частности. Например, Linux Documentation Project (LDP – проект документирования Linux) предлагает доступ к множеству документов по Linux через WWW. Домашняя страница LDP (рис. 5.1), которую можно найти по адресу *http://www.tldp.org*, содержит ссылки на множество других связанных с Linux страниц по всему миру.

Веб-броузеры Linux обычно могут отображать информацию, предоставляемую различными типами серверов, а не только HTTP-серверами, посылающими клиентам страницы HTML. Например, обращаясь к документу по HTTP, вы, возможно, увидите страницу, подобную изображенной на рис. 5.1, то есть с вложенными рисунками, ссылками на другие страницы[1] и т. д. При обращении к документу по FTP вы увидите листинг каталогов FTP-сервера (рис. 5.2). Щелкнув на

[1] Тем не менее и то, и другое, и третье будут компонентами HTML-документа, передаваемого HTTP-протоколом, возможно, с той только разницей, что этот HTML-документ будет сгенерирован динамически. Даже в куда более замысловатых случаях взаимодействий, например построенных на технологии Ajax, все взаимодействие с WWW будет происходить в рамках HTTP. Так что здесь авторы явно погорячились... – *Примеч. науч. ред.*

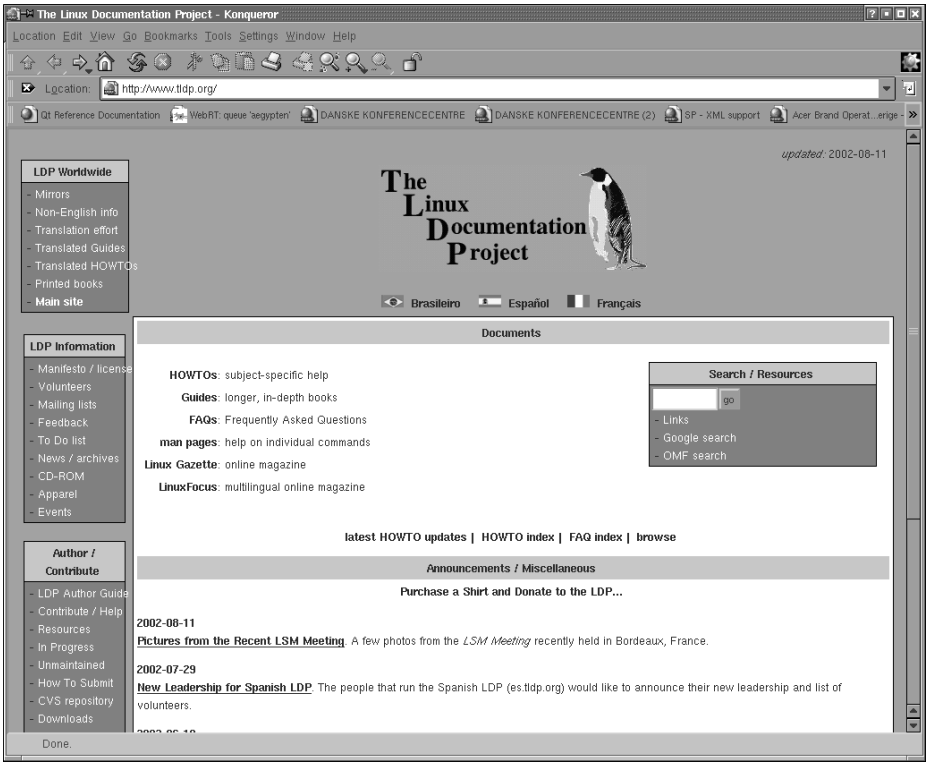


Рис. 5.1. *Домашняя страница Linux Documentation Project (LDP) в WWW*

ссылке в FTP-документе, вы либо загрузите указанный файл, либо увидите содержимое другого каталога.

Ссылка на документ или другой ресурс Сети осуществляется с помощью унифицированного идентификатора ресурса (Uniform Resource Locator), или URL. URL – это имя, уникальным образом идентифицирующее веб-документ, включая имя машины, на которой находится документ, имя файла и используемый для доступа к нему протокол (FTP, HTTP и т. д.). Например, документ «Font HOWTO», описывающий порядок оптимизации использования шрифтов в Linux, имеет следующий URL:

http://www.tldp.org/HOWTO/html_single/Font-HOWTO/index.html

Рассмотрим этот URL. Его первая часть – *http:* – идентифицирует протокол, используемый для доступа к документу, которым в данном случае является HTTP. Вторая часть URL – *//www.tldp.org* – идентифицирует машину, на которой находится документ. Последняя часть URL – *HOWTO/html_single/Font-HOWTO/index.html* – это логический путь к документу на *www.tldp.org*. Он похож на путь к файлу в UNIX и указывает на файл *index.html* в каталоге *HOWTO/html_single/Font-HOWTO*. Поэтому для доступа к «Font HOWTO» надо запустить броузер, указав ему адрес *http://www.tldp.org/HOWTO/html_single/Font-HOWTO/index.html*. Что может быть проще?

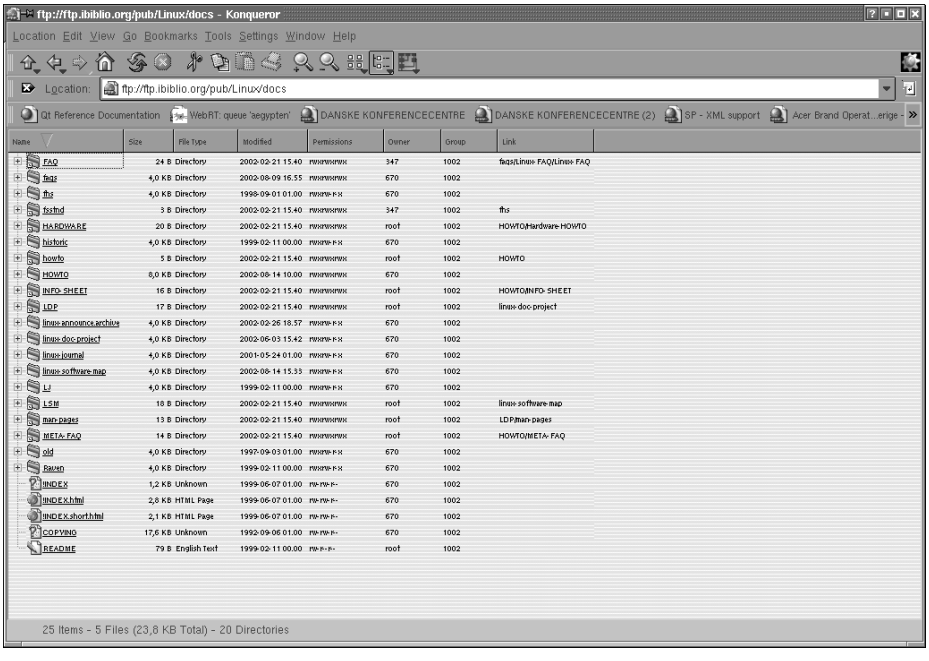


Рис. 5.2. *Отображение содержимого каталога на FTP-сервере в веб-броузере Konqueror*

Оказывается, благодаря некоторым соглашениям, принятым для веб-серверов, это действительно можно сделать проще. Если в качестве последнего элемента URL указывается каталог, сервер поймет, что вам нужен файл *index.html* в этом каталоге.[1] Поэтому вы можете добраться до документа «Font HOWTO» по более короткому URL:

http://www.tldp.org/HOWTO/html_single/Font-HOWTO/

Для доступа к файлу через анонимный FTP-сервер можно использовать URL:

ftp://ftp.ibiblio.org/pub/linux/docs/FAQ

Этот URL вернет список наиболее часто задаваемых вопросов с ответами. Использование этого URL в броузере идентично использованию клиента *ftp* для получения файла вручную.

Чтобы лучше понять Web, нужно просто начать им пользоваться. В следующем разделе мы объясним, как работать с некоторыми броузерами, а затем расскажем, как настроить свою машину для работы в качестве веб-сервера, предоставляющего документы в Web.

---

[1] Это не совсем точно: в настройках веб-сервера устанавливается имя документа по умолчанию, и чаще всего это имя именно *index.html*, хотя это совсем не обязательно. Достаточно часто встречается, например, установка по умолчанию *home.html*. – *Примеч. науч. ред.*

Конечно, для того чтобы войти в Web, система должна иметь прямое соединение с Интернетом (по Ethernet или PPP). В следующих разделах мы предполагаем, что вы уже настроили TCP/IP на своей системе и можете успешно пользоваться такими клиентами, как *ssh* и *ftp*.

Использование Konqueror

Konqueror – это один из самых популярных броузеров для Linux. Он поддерживает JavaScript и Java, может выполнять подключаемые модули Firefox (что позволяет добавлять такие функции, как просмотр презентаций Flash) и интегрирован в окружение рабочего стола KDE, описанное в разделе «K Desktop Environment» главы 3. На самом деле при установке KDE в качестве составной части системы устанавливается Konqueror. В разделе, посвященном KDE, уже говорилось, как с помощью Konqueror просматривать локальные файлы документации. Теперь мы отправимся с ним в Web.

Работа с Konqueror обычно самоочевидна, но если вы хотите познакомиться с ним глубже, зайдите с помощью Konqueror на сайт *http://www.konqueror.org*.

Ниже предполагается наличие Linux-машины, подключенной к сети, с работающей системой X и броузером Konqueror. Как отмечалось ранее, машина должна быть настроена на работу с TCP/IP и такими клиентами, как *ssh* и *ftp*.

Запустить Konqueror просто. Выполните команду:

```
eggplant$ konqueror url
```

где *url* – это полный веб-адрес, или URL, документа, который вы хотите увидеть. Если вы не укажете URL, Konqueror по умолчанию должен вывести рекламную заставку, как показано на рис. 5.3.

При запуске Konqueror из KDE можно просто нажать Alt+F2, чтобы открыть окно ввода команд *minicli* и ввести URL. В результате запустится Konqueror, который откроет указанный вами URL.

Мы предполагаем, что у читателя есть некоторый опыт работы с веб-броузером на какой-либо компьютерной системе, поэтому не станем останавливаться на элементарных вещах, а отметим некоторые детали, специфичные для Linux.

Имейте в виду, что доступ к документам в Web иногда может быть медленным. Это зависит от скорости сетевого соединения между вами и сервером, а также от сетевого трафика. Иногда при большой загруженности веб-сайты просто отклоняют новые соединения. В этом случае Konqueror выводит соответствующее сообщение об ошибке. В нижней части окна Konqueror выводится информация о состоянии соединения, которая меняется в процессе передачи данных, эмблема KDE в правом верхнем углу оживает. Щелчок на эмблеме возвращает к домашней странице Konqueror.

При переходах по ссылкам в Konqueror каждый документ сохраняется в *журнале*, и к нему можно вернуться через меню Go (Перейти). Нажатие кнопки Back (Назад) (со стрелкой, направленной влево) в верхней части окна Konqueror возвращает к документам, которые просматривались ранее. Аналогично кнопка Forward (Вперед) перемещает вперед по уже посещенным ссылкам.

Рис. 5.3. *Рекламная заставка Konqueror*

Кроме того, в Konqueror имеется боковая панель, где выводится список посещавшихся ранее веб-сайтов. Это очень удобно, когда возникает потребность вернуться на сайт, который уже посещался некоторое время тому назад – не слишком давно, чтобы адрес сайта все еще оставался в журнале, но достаточно давно, чтобы забыть его название. В боковой панели журнала адреса (URL) сортируются по именам сайтов. Если у вас в броузере боковая панель не видна, попробуйте нажать клавишу F9 или выбрать пункт главного меню Window (Окно)→Show Navigation Panel (Показать панель навигации). В боковой панели имеется несколько вложенных панелей, при этом в каждый момент времени видимой может быть только одна из них; нужная нам панель имеет вкладку с изображением часов – щелкните на ней мышью, чтобы увидеть список ранее посещавшихся сайтов.

Для часто посещаемых сайтов (или URL) можно делать *закладки*. При просмотре документа, к которому вы, возможно, захотите вернуться в будущем, выберите в меню пункт Bookmarks (Закладки)→Add Bookmark (Добавить закладку) или просто нажмите комбинацию Ctrl+B. Чтобы просмотреть список закладок, выберите пункт меню Bookmarks (Закладки). Щелчок на любой из закладок приведет вас к соответствующему документу в Web. Наконец, закладки могут отображаться постоянно в боковой панели, для чего нужно щелкнуть на вкладке с изображением желтой звездочки. И, разумеется, Konqueror обладает достаточно широкими возможностями по управлению закладками. Просто выберите пункт меню Bookmarks (Закладки)→Edit Bookmarks (Изменить закладки) и вперед!

С помощью боковой панели можно перейти в свой домашний каталог, открыть список установленного оборудования, просмотреть журнал истории сеанса работы

и многое другое. Попробуйте поработать с Konqueror, и вы найдете в нем много полезных функций.

Помимо боковой панели Konqueror имеет еще одну особенность, которая может значительно облегчить ваши путешествия в Web, – это так называемые *вкладки*. Впервые вкладки появились в броузере Mozilla (будет описан далее в этой главе), распространяемом с открытыми исходными текстами, затем Konqueror взял механизм вкладок за основу и добавил к нему множество полезных особенностей. Например, во время чтения веб-страницы, содержащей интересную ссылку, по которой хотелось бы сходить немного погодя, теперь возможно не прерывать чтение документа, а просто щелкнуть правой кнопкой мыши на ссылке и выбрать в контекстном меню пункт Open in New Tab (Открыть в новой вкладке). В результате будет создана новая вкладка с названием страницы в заголовке, причем текущая страница также останется открытой. Закончив чтение текущей страницы, можно будет перейти к просмотру той, что была открыта в процессе чтения. Так как все страницы находятся на вкладках в единственном окне броузера, пространство рабочего стола не загромождается, и найти требуемую страницу не составляет никакого труда.[1] Чтобы закрыть вкладку, достаточно нажать кнопку с изображением красного крестика, расположенную в правой части панели с вкладками.

Как упоминалось ранее, можно обратиться к новому URL, указав его в качестве аргумента при запуске `konqueror`. Однако можно просто ввести URL в адресной строке в верхней части окна Konqueror. Адресная строка поддерживает функцию автозавершения: если начать ввод адреса, который вы уже посещали, Konqueror автоматически покажет его и даст возможность выбрать. Закончив ввод URL (с помощью функции автозавершения или без нее), нужно нажать Enter, и вы получите требуемый документ.

Konqueror – это мощное приложение с множеством настраиваемых параметров. Для настройки его поведения нужно выбрать пункт меню Settings (Настройки)→ Configure Konqueror (Настроить Konqueror). Очень интересные настройки можно найти в разделах Web Behavior (Поведение Веб) и Web Shortcuts (Сокращения Веб). В разделе Cookies можно установить для каждого домена, будут ли от него приниматься файлы cookies, и просмотреть те cookies, которые уже есть на компьютере. Сравните это с другими броузерами, которые прячут файлы cookies где-то так глубоко, что их трудно посмотреть (или даже невозможно без специальных программ!).

Наконец, стоит отметить одну особую функцию. Веб-броузеры регистрируются на серверах с использованием так называемой строки идентификации броузера, которая может содержать любой текст, но обычно хранит название и версию веб-броузера, а также название и версию операционной системы хоста. Некоторые недостаточно грамотные веб-мастеры создают разные веб-страницы (или вообще

---

[1] Но каждая открытая вкладка «отбирает» свободную память компьютера, причем изрядно (правда, не больше, чем если бы мы открыли этот URL в новом окне), поэтому открыв 20–30–40 вкладок, мы можем полностью исчерпать ресурсы компьютера по памяти. В Windows это чаще всего завершается крахом всей системы, а в Linux – крахом «всего лишь» приложения, того же Konqueror, но при этом «рушатся» все вкладки, начиная с самой первой. – *Примеч. науч. ред.*

ничего не создают!), если веб-броузер не идентифицирует себя как Internet Explorer, поскольку полагают, что Internet Explorer – единственный веб-броузер, который в состоянии отображать содержимое их веб-сайта.[1] Но если перейти в раздел Browser Identification (Идентификация броузера), можно обмануть веб-сервер, прикинувшись другим броузером, которому этот веб-сервер не стыдится выдавать документы. Для этого нужно щелкнуть на кнопке New (Добавить), выбрать имя домена, к которому вы хотите получить доступ, и либо ввести свою строку идентификации, либо выбрать одну из предопределенных.

Другие броузеры

Konqueror – не единственный броузер, способный читать документы Web. В Linux имеется еще один броузер – Firefox, являющийся наследником Mozilla[2], который, в свою очередь, зарождался как открытая версия броузера Netscape Navigator, благодаря чему очень многие познакомились с Web. Если в вашем дистрибутиве Firefox отсутствует, его можно загрузить со страницы *http://www.mozilla.org/products/firefox/*.[3] Функции Firefox вполне аналогичны возможностям Konqueror, и все, что можно делать в одном броузере, можно делать и в другом. Главное преимущество Konqueror перед Firefox состоит в том, что он интегрирован в окружение рабочего стола KDE, и это весьма немаловажно, если, конечно, вы пользуетесь KDE. Сильной стороной Firefox является его интеграция с нестандартными технологиями, такими как Flash.[4] Кроме того, в Firefox реализована очень удобная функция блокирования всплывающих окон. Эта функция может быть настроена так, что она будет или блокировать все всплывающие окна (и никогда не сообщать об этом), или постоянно пропускать всплывающие окна с определенных сайтов (в которых может содержаться важная информация, например о вашем банковском счете), или позволит всплывать окнам лишь один раз.

Firefox обладает одной важной характеристикой, которую часто упускают из виду: поддержка модулей расширения. Если выбрать пункт главного меню Tools (Сервис)→Extensions (Дополнения), на экране появится диалог со списком установленных модулей расширения. Скорее всего, изначально этот список будет пуст (если распространитель дистрибутива или ваш администратор не установили некоторые из них заранее). Щелкнув на ссылке Get More Extensions (Загрузить

---

[1] Если веб-сайт можно просматривать лишь одним броузером или на нем стоит пометка «оптимизирован для броузера X», то с него нужно бежать, сдерживая возмущение от такой вопиющей некомпетентности веб-мастера.

[2] Точное название именно «Mozilla Firefox», т. к. в рамках того же семейства параллельно развивается и другая независимая линия – Mozilla SeaMonkey, являющаяся «чистым» наследником броузера Netscape Navigator. На текущий момент последние версии – Mozilla Firefox 2.0 и Mozilla SeaMonkey 1.1. – *Примеч. науч. ред.*

[3] Русскоязычные ресурсы: *http://ru.www.mozilla.com/ru/firefox/about/*, *http://www.mozilla-russia.org/*. – *Примеч. науч. ред.*

[4] Самым большим (и не упомянутым) достоинством Mozilla Firefox является то, что он реализован практически «в одной ипостаси» чуть ли не для всех существующих операционных систем: Windows, Linux, QNX, SunSolaris и других. Вы можете садиться за совершенно новую систему... и продолжать работу со знакомым Mozilla Firefox! – *Примеч. науч. ред.*

расширения), вы получите длинный список расширений для Firefox. По умолчанию в окне броузера появятся два списка – наиболее популярных и самых новых расширений, но, потратив некоторое время на просмотр всех категорий, можно будет обнаружить в них немало интересных и нужных расширений.

Здесь мы отметим два расширения, которые показались нам особенно интересными. Первое из них называется Adblock. После установки этого расширения в окне броузера появится полупрозрачный значок, напоминающий закладку, напротив тех элементов веб-страниц, которые по определенным критериям попадают в категорию рекламных баннеров. Просто щелкните на значке, затем щелкните на кнопке OK в появившемся диалоге (или отредактируйте URL блокируемого элемента, чтобы, например, сделать его более универсальным) и наслаждайтесь просмотром веб-страниц без рекламных баннеров. Постепенно вы научитесь улучшать качество шаблонов блокирования настолько, что никогда не будете видеть баннеры во время путешествий в Web.

Другое, не менее интересное расширение называется ForecastFox. Оно позволяет выбирать различные регионы по всему земному шару и отображать погоду в них в виде небольших значков в строке состояния (или в другом месте по вашему выбору). При наведении указателя мыши на эти значки будут появляться небольшие подсказки с более подробной информацией.

Как и с Konqueror, вам определенно стоит поработать некоторое время с Firefox, чтобы изучить все его возможности. Благодаря своим характеристикам, таким как безопасность и удобство, он претендует на роль самого популярного веб-броузера.[1]

Еще один универсальный броузер – w3m. Это текстовый броузер, поэтому, работая с ним, вы не увидите графику. Однако данный факт делает его быстрым, что может показаться вам удобным. Им можно пользоваться даже без X Window System. Кроме того, когда возникает необходимость сохранить веб-страницу в виде простого текстового файла, w3m зачастую обеспечивает гораздо более качественное форматирование, чем другие броузеры, поскольку отображение текста – его основное предназначение. Еще один броузер, финансируемый за счет рекламы и приобретший большую популярность в последнее время, – Opera. Наконец, для тех, кто никогда не покидает Emacs, существует Emacs/W3 – полноценный веб-броузер, которым можно пользоваться прямо из Emacs или XEmacs.

Обмен мгновенными сообщениями

Самые разнообразные формы прямого общения пользователей компьютеров по Сети известны уже на протяжении нескольких десятилетий, тем не менее с ростом числа пользователей Интернета все большую популярность стала завоевывать технология обмена мгновенными сообщениями (Instant Messaging, IM). AOL Instant Messenger (AIM), Yahoo! Messenger и MSN Messenger – это лишь два из многих вариантов реализации этой технологии. Каждая подобная служба предоставляет свое собственное клиентское программное обеспечение (и всячески способствует тому, чтобы вы использовали именно его, поскольку таким

[1] По всем статистистическим данным на сегодня он является самым используемым броузером в мире. – *Примеч. науч. ред.*

способом они смогут обеспечить себе возможность передачи рекламных объявлений). Но тем не менее любой может получить доступ к самым популярным системам обмена мгновенными сообщениями, используя открытые программы, такие как Gaim, Kopete и большое разнообразие Jabber-клиентов. Эти программы являются полноценными клиентами, они обладают массой уникальных особенностей, а по своим функциональным возможностям опережают программы, навязываемые коммерческими службами (хотя, надо признать, в открытых программах отсутствуют некоторые привлекательные особенности коммерческих программ).

К сожалению, каждая коммерческая служба обмена мгновенными сообщениями стремится использовать свой собственный протокол передачи данных, и ни один из протоколов не может использоваться для работы с другими службами. Такое разнообразие несовместимых между собой протоколов явилось следствием желания каждого поставщика вынудить людей использовать его клиентское программное обеспечение и получать его рекламные объявления. Поскольку сама услуга предоставляется бесплатно, такое желание вполне можно объяснить стремлением возместить затраты за счет рекламных акций. По меньшей мере одна популярная служба обмена сообщениями (Yahoo!) предлагает реализацию своего клиента для Linux, причем очень даже неплохого.

Но уже настало время, когда цифровые видеомагнитофоны «научились» пропускать телевизионные рекламные ролики. Аналогичным образом открытое программное обеспечение позволяет легко и просто обмениваться мгновенными сообщениями, не выплескивая вам в лицо прогнозы погоды или фотографии попзвезд. Самое важное, пожалуй, заключается в том, что клиенты с открытыми исходными текстами позволяют использовать одну и ту же программу для работы с различными службами обмена сообщениями. Благодаря этому отпадает необходимость иметь несколько программ и выполнять настройку каждой из них в отдельности. Есть надежда, что, в конечном счете, коммерческие поставщики услуг могут пойти на уступки и принять за основу расширяемый протокол присутствия и передачи сообщений (Extensible Messaging and Presence Protocol, XMPP), который имел немного странное название Jabber, до того как стал стандартом (точнее, набором документов RFC, выпущенных комитетом IETF), а пока пользуйтесь клиентами, обладающими поддержкой нескольких протоколов.

Все эти клиенты имеют интуитивный интерфейс, но есть некоторые хитрости, о которых вам следует знать. В этом разделе мы расскажем о Gaim – наиболее популярной среди пользователей Linux программе обмена сообщениями. Второе место по популярности можно отдать Kopete – клиенту из KDE.

Программа Gaim входит в состав большинства дистрибутивов Linux, и ее обычно можно найти где-нибудь под пунктом меню, за которым находятся программы для работы в Интернете. Если вам встретится пункт под названием, например, «клиент обмена мгновенными сообщениями», скорее всего, за ним будет находиться Gaim (или Kopete). Обычно, если программа Gaim установлена, ее можно запустить командой *gaim* из командной строки. Если она не была установлена, вы без труда сможете сделать это самостоятельно, загрузив программу с сайта *http://gaim.sourceforge.net*.

Описание, следующее ниже, относится к версии 1.2.1 для Linux. Более новая версия клиента, выход которой ожидался, когда писались эти строки, скорее всего,

будет иметь систему меню, отличающуюся своим содержимым, и другой интерфейс, но будет предоставлять как минимум те же функции.

Первоначальная настройка

В этой книге не будет говориться о том, как получить учетную запись в системе обмена мгновенными сообщениями. Для этого вам нужно посетить веб-сайт приглянувшейся системы и просто следовать пошаговой инструкции, которую вы без труда там найдете. После того как учетная запись будет получена (это тяжкий труд – подыскать себе псевдоним, который еще никем не занят), можно приступать к настройке Gaim. Сразу после запуска программа должна отобразить окно со списком учетных записей (рис. 5.4). Если списка учетных записей не появилось, но было выведено окно, где должен размещаться список собеседников, нажмите комбинацию Ctrl+A или выберите пункт меню Tools (Сервис)→Accounts (Учетные записи).

Щелкните на кнопке Add (Добавить) и заполните форму информацией об учетной записи:

Протокол

Здесь должна быть выбрана система обмена мгновенными сообщениями, которая будет использоваться. По умолчанию предлагается наиболее популярная AIM/ICQ, но если предполагается использовать другую систему, просто выберите ее из раскрывающегося списка. Среди возможных вариантов будет предложена и система IRC, таким образом, Gaim может использоваться для участия в конференциях на сайтах IRC, которые очень популярны среди пользователей и разработчиков Linux.

Идентификатор пользователя

Это имя учетной записи, которое будет использовано для регистрации.

Пароль

Пароль выбирается в процессе получения учетной записи.

Псевдоним

Это имя, под которым вы будете видеть себя самого в чате, но оно никак не влияет на то, что будут видеть ваши собеседники.

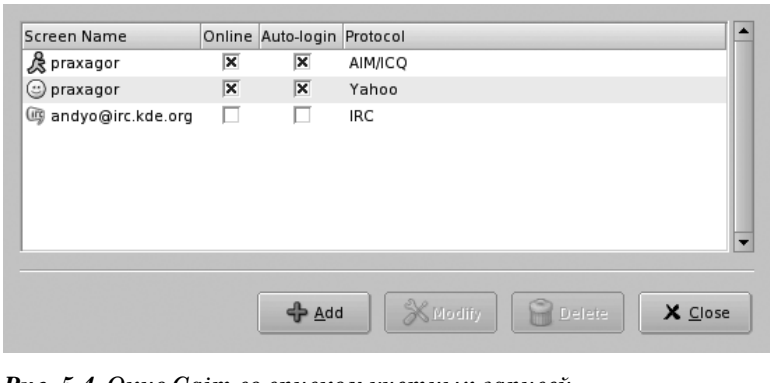


***Рис. 5.4.** Окно Gaim со списком учетных записей*

В этом диалоге имеется еще много настраиваемых параметров. Например, если необходима настройка соединения через прокси-сервер, определить протокол соединения можно, щелкнув на метке Show more options (Показать больше параметров). Доступ к этому (и многим другим) параметру можно получить и из окна со списком контактов, выбрав пункт меню Tools (Сервис)→Preferences (Настройки) или просто нажав комбинацию Ctrl+P. Обратите внимание: настройки, произведенные в диалоге, вызванном с помощью пункта меню Preferences (Настройки), определяют значения параметров по умолчанию для всех учетных записей, которые можно переопределить в настройках каждой отдельной учетной записи.

Если у вас имеется свой персональный компьютер или ноутбук, который имеет подключение к Интернету, очень удобно настроить Gaim так, чтобы программа запоминала пароль и выполняла процедуру регистрации автоматически. Но если вы редко пользуетесь системой обмена мгновенными сообщениями или компьютер доступен не только вам, тогда можно оставить значения этих параметров по умолчанию.

Пока мы не сделали ничего особенного, чтобы придать вам индивидуальные черты в Интернете (это будет сделано ниже, в разделе «Дополнительные настройки»), но достаточно, чтобы дать возможность общаться с другими людьми.

По окончании настройки необходимо сохранить информацию об учетной записи. По возвращении в окно со списком учетных записей установите флажок в поле Online (В сети). Если в это время компьютер соединен с Интернетом, программа запустит процедуру регистрации, и вы сможете двинуться дальше. Если процедура регистрации потерпит неудачу, щелкните на кнопке Modify (Изменить) и проверьте еще раз, все ли параметры были настроены правильно. Особое внимание обратите на правильность выбора протокола. Попробуйте повторно ввести пароль.

Общение

С этого момента в работе с Gaim нет ничего хитрого. Большинство людей предпочитают общаться только со своими знакомыми и только после явного добавления их к списку учетных записей, который называется *списком контактов*. Большинство систем обмена сообщениями сохраняют эту информацию у себя, поэтому, если вы уже добавили своих приятелей в другом клиенте, они появятся в списке контактов Gaim.

Чтобы добавить новый контакт, вызовите меню Buddies (Контакты). Сначала добавьте несколько групп контактов, например: «Работа», «Семья» и «Политические споры». (Осознание удобств, которые дают группы, придет к вам после нескольких недель работы с Gaim, когда вы поймете, с каким количеством людей приходится общаться. Некоторые из авторов этой книги болтают в чате с членами семьи, которые находятся в соседней комнате. Неужели это лучше, чем личная беседа?)

После этого добавьте в группы своих собеседников. Как найти собеседников? Это как с процедурой приобретения учетной записи – «вне полосы пропускания» – компьютерный термин, который можно трактовать так: «вы сами решаете, делать это или нет, и нас совершенно не интересует, как вы это сделаете». Большинство обмениваются именами учетных записей по электронной почте или передают их друг другу в виде записок на бумаге. Но есть один удобный метод поис-

ка в системе AIM/ICQ – через пункт меню Tools (Сервис)→Account Actions (Действия с учетными записями)→Search for Buddy by Email (Искать пользователя по e-mail).

Чтобы запустить сеанс связи, дважды щелкните на каком-либо из контактов в списке тех, кто уже зарегистрировался в системе. Чтобы начать общение сразу с несколькими собеседниками, которые пользуются той же самой системой обмена мгновенными сообщениями, выберите пункт меню Buddies (Контакты)→ Join a Chat (Присоединиться к чату). Здесь вам будет предложено выбрать используемую службу и имя по вашему желанию. Затем предложите своим собеседникам войти, для каждого из них щелкая на кнопке Invite (Пригласить), выбирая собеседника из раскрывающегося меню и вводя текстовое сообщение о том, что он или она приглашаются к общению. Вы можете общаться с собеседниками из разных систем IM в разных чатах (например, AOL и MSN), но вы не можете объединить собеседников, пользующихся разными системами IM в одном чате, потому что каждая система использует свой собственный протокол.

Одной из самых ценных особенностей обмена мгновенными сообщениями, которая превращает его в инструмент настоящего бизнеса, а не только приятного времяпрепровождения, является способность сохранять текст чатов. Благодаря этому можно спустя некоторое время обратиться к своим высказываниям (к обещаниям, которые вы, возможно, сделали). Для этого, находясь в чате, выберите пункт меню Conversations (Беседа)→Save As (Сохранить как) и сохраните текст в формате HTML. Сохранено будет то, что к настоящему моменту уже имеется в окне, если же чат еще не закончен и появится новый текст, который необходимо будет сохранить, придется повторно выполнить операцию сохранения. Может оказаться удобным включить автоматическое журналирование всех чатов или мгновенных сообщений по умолчанию. Сделать это можно через пункт Logging (Вести журнал) в меню Preferences (Настройки), но при этом вы должны быть готовы, что наряду с полезной информацией будет сохраняться большой объем «словесного мусора».

Формат HTML плохо подходит для нужд журналирования, но он более удобочитаем и в нем проще отыскивать интересующие вас куски текста. Если временные метки в каждом сообщении не имеют для вас никакого значения, просто отключите их в раскрывающемся меню Options (Параметры).

Маленькие кнопки с изображением символа A отражают различные типы форматирования текста (курсив, полужирный и даже цветной), которые можно использовать: с помощью мыши выделите участок текста, который требуется изменить, и щелкните на кнопке. Вместо мыши и кнопок на экране можно использовать быстрые комбинации клавиш, например, чтобы сделать шрифт выделенного текста жирным, нажмите Ctrl+B, курсивным – Ctrl+I, с подчеркиванием – Ctrl+U, перечеркнутым – Ctrl+S. Если некоторый текст был выделен и вы хотите отменить его форматирование, просто щелкните на соответствующей кнопке или нажмите клавишу Ctrl еще раз.

Задолго до появления систем обмена мгновенными сообщениями пользователи систем обмена текстовыми сообщениями, таких как электронная почта, обмен новостями и IRC (Internet Relay Chat), проявили изобретательность и придумали маленькие текстовые изображения рожиц, такие как :-) и :-<, которые известны под названием «смайлики». Переместившись в графическую среду, клиенты IM добавили возможность отображения графических смайликов. Если вам

не нравится стандартный набор смайликов, входящий в состав Gaim по умолчанию, его можно заменить другим. Для этого зайдите на сайт Gaim. (Щелкните на ссылке Themes (Темы) с правой стороны главной страницы.) Загрузите архив с набором смайликов, которые вам покажутся наиболее интригующими (к сожалению, выбор всей темы придется делать по одному-единственному изображению), и распакуйте его в подкаталог *smileys* с файлами настроек Gaim, обычно это *~/.gaim/smileys*.

При вводе адреса URL в чате он автоматически преобразуется в ссылку. Однако, если вы хотите получить более изысканный вариант форматирования текста, например обозначить ссылку словами «Моя домашняя страница», щелкните на кнопке с изображением железной цепочки. После этого введите адрес URL и слова, которые в тексте сообщения будут преобразованы в ссылку на этот URL. Отправить файл своему собеседнику так же просто, как выбрать пункт меню Conversations (Беседа)→Send File (Отправить файл). Однако файл не будет передаваться до тех пор, пока ваш собеседник не согласится принять его.

Дополнительные настройки

Вы не можете выйти из дома без своей тени, подобным же образом ваше общение с IM будет неполным без дополнительной настройки некоторых характеристик, которые будут подчеркивать вашу индивидуальность:

- Информация для собеседников (в некоторых других клиентах известна как *профиль*) – текст, в свободной форме описывающий вас.
- Маленькое изображение.
- Необычный набор фраз для автоответчика, который будет сообщать вашим собеседникам ваш статус, – тема жарких споров в мире исследования сетевых взаимодействий, называемая *присутствием*.

Кроме того, в этом разделе мы обсудим некоторые другие настройки, которые вы наверняка сочтете полезными, включая способ понять, чем занимаются ваши собеседники.

Информацию для собеседников можно ввести, выбрав пункт меню Tools (Сервис)→Account Actions (Действия с учетными записями)→Set User Info (Настроить информацию о пользователе). Обратите внимание: эта информация (и все настройки) будет относиться только к клиенту Gaim, который в настоящий момент используется. Если вы будете использовать Gaim для работы в другой системе обмена мгновенными сообщениями или будете использовать другой клиент IM, вам придется снова ввести всю информацию, чтобы сделать ее доступной для собеседников. Поэтому, чтобы сократить объем вводимого текста, подумайте о том, чтобы добавить ссылку на домашнюю страницу, где собеседники смогут узнать о вас подробнее.

Как и другие клиенты IM, Gaim позволяет присоединить графическое изображение к учетной записи, которая будет отображаться у ваших собеседников в списке контактов и при разговоре. Чтобы присоединить картинку к учетной записи, при создании новой учетной записи в диалоге Add Account (Добавить учетную запись) или при настройке существующей в диалоге Modify Account (Изменить учетную запись) щелкните на кнопке Open (Открыть) рядом с меткой Buddy icon (Значок пользователя) и найдите в файловой системе подходящее изображение.

Можно просто открыть нужный каталог в менеджере файлов вашего рабочего стола и перетащить нужный файл мышью в окно Modify Account (Изменить учетную запись). Gaim поддерживает многие известные графические форматы, включая JPEG, GIF и PNG. В зависимости от наличия поддержки в библиотеках GTK+, Gaim автоматически преобразует формат выбранного файла в формат, который сможет принять система обмена мгновенными сообщениями, если в этом возникнет необходимость.

В AIM налагаются весьма существенные ограничения на размеры изображения, и Gaim, к сожалению, не сможет сообщить вам, что ограничения были превышены. Кроме того, для большинства систем IM необходимо, чтобы изображение имело геометрическую форму как можно ближе к квадрату, потому что в противном случае рисунок может быть вытянут по одной из осей и выглядеть весьма незавидно. Для приведения изображений в соответствие предъявляемым требованиям можно использовать отличный графический редактор GIMP (глава 9).

Чтобы добавить оригинальные сообщения для автоответчика, выберите пункт меню Tools (Сервис)→Preferences (Настройки)→Away messages (Сообщения об отсутствии) и щелчком на кнопке Add (Добавить) откройте диалог, в котором вы сможете добавлять и сохранять свои собственные сообщения. (Или Tools (Сервис)→Away (Статус)→New Away message (Новое сообщение об отсутствии).) Для каждого сообщения определяются: название, которое отображается в меню, и, собственно, текст сообщения, который будут видеть ваши собеседники.

Когда вы будете покидать свое место, можно выбрать наиболее подходящее сообщение для автоответчика из меню Tools (Сервис)→Away (Статус)→Custom (<и далее по выбору>), это очень удобно для ваших собеседников. Многие нередко забывают, что создание новых сообщений – это еще не все, поскольку Gaim автоматически выбирает одно из сообщений при длительном отсутствии активности за терминалом. Мы советуем заменить надоедливое сообщение по умолчанию (если вам оно не кажется надоедливым, просто посмотрите на него) вашим собственным сообщением. Сделать это можно из диалога Preferences (Настройки), открыть который можно нажатием комбинации Ctrl+P. Выбрав элемент Away/Idle (Отсутствие/Бездействие), можно переопределить сообщение автоответчика по умолчанию, а также время отсутствия активности за терминалом, которое должно пройти, прежде чем это сообщение появится.

Если сообщение автоответчика, сообщающее о вашем отсутствии, было установлено по таймеру через определенное время, Gaim автоматически заменит его сообщением Available (Активен), когда обнаружит движение указателя мыши или нажатие клавиш на клавиатуре. Если же сообщение об отсутствии было установлено явно, вы точно так же явно должны обозначить свое возвращение, выбрав пункт меню Tools (Сервис)→Away (Статус)→Back (<предыдущее сообщение>). Сообщение с заголовком Available (Готов пообщаться), свидетельствующее о вашем присутствии за терминалом, может быть изменено через пункт меню Tools (Сервис)→Account Actions (Действия с учетными записями)→Set Available message (Установить заголовок «Готов пообщаться»).

В процессе набора текста Gaim автоматически выполнит проверку правописания и подчеркнет слова с орфографическими ошибками. Поскольку при неформальном общении всегда присутствует атмосфера свободы и непринужденности, где точная проверка правописания не так важна, эта функциональная возможность

выглядит несколько излишней. Она прекрасно справляется с возложенными на нее задачами и отлично адаптируется под национальные особенности (то есть язык и национальность, которые были выбраны при установке дистрибутива), но при желании может быть выключена в диалоге Preferences (Настройки), пункт Message (Сообщение)→Text box (Текстовое окно).

В Gaim присутствует гораздо более полезная особенность для тех, кто еще плохо освоил клавиатуру, – замена текста. Данная функциональность реализована в виде модуля расширения, который можно включить в диалоге с настройками. Для этого нужно выбрать пункт Plugins (Модули) и включить модуль Text replacement (Замещение текста). После этого можно вводить текст в виде сокращений общих фраз. Например, один из авторов этой книги определил сокращение `newrl`, которое разворачивается в строку «Running Linux, 5th Edition», чтобы упростить вставку ссылок на эту книгу. Сокращения должны вводиться как отдельные слова, чтобы Gaim мог распознавать и замещать их.

Выше мы уже говорили о том, каким образом ваши собеседники смогут узнать о вашем отсутствии за терминалом. Аналогичным образом Gaim может отображать информацию об отсутствии ваших собеседников, но по умолчанию сообщение об уходе или приходе собеседника не выскакивает автоматически (как это сделано во многих других клиентах IM). Добавить такую возможность можно, установив модуль расширения guifications. Загрузите его с сайта *http://guifications.sourceforge.net*, установите и включите в диалоге Preferences (Настройки), в разделе Plugins (Модули).

Но даже без этого модуля в распоряжении пользователя имеется немало средств наблюдения за присутствием собеседников. Можно настроить Gaim так, что он будет уведомлять вас: когда подключится определенный собеседник, когда он отключится, когда он отошел, когда он вернулся и т. д. Таким образом, имеется возможность выбрать конкретного пользователя, об изменении статуса которого вы будете получать уведомления. Это может потребоваться, когда, например, вам срочно нужно переговорить с этим человеком, чтобы обсудить важную тему. Подобное слежение выполняется с помощью механизма правил слежения за пользователями.

Чтобы воспользоваться этой особенностью, выберите пункт меню Tools (Сервис)→Buddy Pounce (Слежение за пользователями)→New Buddy Pounce (Новое правило). В диалоге, который после этого появится, можно определить пользователя, за которым будет вестись слежение, и указать, какие изменения в статусе требуется отследить, а также какое уведомление и как должно выводиться. Ваш собеседник даже не будет догадываться о том, что вы отслеживаете его присутствие, если, конечно, вы не настроите возможность отправки сообщения. Данная особенность может использоваться для вывода некоторого сообщения на экране вашего собеседника, например: «Пожалуйста, позвони домой, как появишься», как только изменится информация о его присутствии.

6

Клиенты электронной почты

Современные программы чтения электронной почты обладают графическим интерфейсом и в своем развитии имеют тенденцию к унификации предлагаемых возможностей и интерфейса. Помимо доставки электронной почты, большинство из них позволяют сопровождать адресные книги и содержат в себе календари. Обычно почтовые клиенты предоставляют возможности чтения лент новостей, которые являются одним из самых старых способов распространения информации в Сети и до сих пор могут служить одним из самых оперативных источников новостей (если, конечно, вам удастся отыскать новостные группы, не замусоренные коммерческой рекламой).

Одним из самых популярных клиентов электронной почты является Evolution, который был описан в главе 3. В этой главе мы продемонстрируем некоторые другие программы чтения электронной почты и приемы работы с ними, что поможет вам работать более продуктивно и решать такие задачи, как доставка электронной почты с сервера в локальную систему с помощью *fetchmail* и защита почтовых отправлений с помощью шифрования.

Linux поддерживает и довольно старые консольные программы для работы с электронной почтой. Например, Elm и Pine – консольные клиенты электронной почты, которые сумели удержаться на достаточно высоком уровне и вполне соответствуют соглашениям, принятым в современном мире электронной почты, таким как отображение на экране содержимого файлов различных типов и переход по ссылкам URL. Некоторым нравится проверенная временем программа `mail`, но обычно она вызывается из сценариев автоматической отправки электронной почты. Эти, уже устаревшие, программы здесь обсуждаться не будут.

Теперь, наверное, есть смысл указать на существующие различия между *почтовым клиентом* (mail user agent, MUA) и *агентом передачи почты* (mail transport agent, MTA). Программа, с которой вы взаимодействуете, читая или отправляя письма электронной почты, является почтовым клиентом, программы, которые описываются в этой главе, – это почтовые клиенты. Агент передачи почты – это программное обеспечение, которое осуществляет прием почты из Интернета и доставляет ее в почтовый ящик пользователя. В качестве примера агента передачи почты можно назвать программу Postfix, которая будет описана в главе 23 «Транспортировка и обработка сообщений электронной почты».

Почтовый клиент KMail

KMail – очень дружественная к пользователю и богатая функциями почтовая программа, поставляемая с KDE и хорошо интегрирующая почту с другими утилитами. Например, если в полученном сообщении есть ссылка на веб-страницу, можно щелкнуть на ней, и тогда появится веб-броузер KDE Konqueror, который отобразит эту страницу. Если же сообщение содержит вложенный файл MP3, можно щелкнуть на нем, и он будет воспроизведен проигрывателем MP3, входящим в KDE. На рис. 6.1 приводится внешний вид окна программы KMail.

В KMail есть масса функций и настроек, но мы расскажем только о тех из них, которые позволят вам быстро начать работу, а глубже изучить KMail вы можете самостоятельно. Как видно из рис. 6.2, окно KMail делится по умолчанию на три части. В левой части показано дерево папок (при первом запуске будут, конечно, видны только папки, устанавливаемые по умолчанию). Справа в верхней части находится список сообщений в текущей выделенной папке, а справа в нижней части – текущее выделенное сообщение. Распределить пространство между этими частями можно иначе, если перетащить разделительную линию между ними. В последних версиях KMail есть еще четвертая часть, с помощью которой можно заглянуть глубже в структуру отдельного сообщения и показать компоненты MIME, из которых оно состоит. Однако по умолчанию такое представление выключено, поскольку не требуется большинству пользователей.

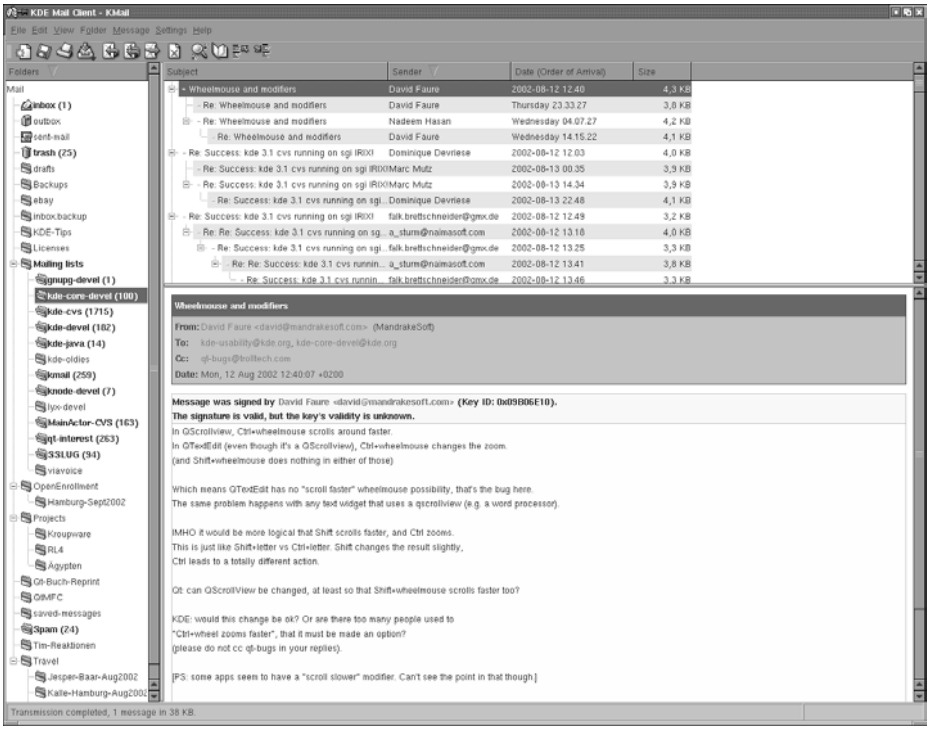


Рис. 6.1. *Почтовая программа KMail*

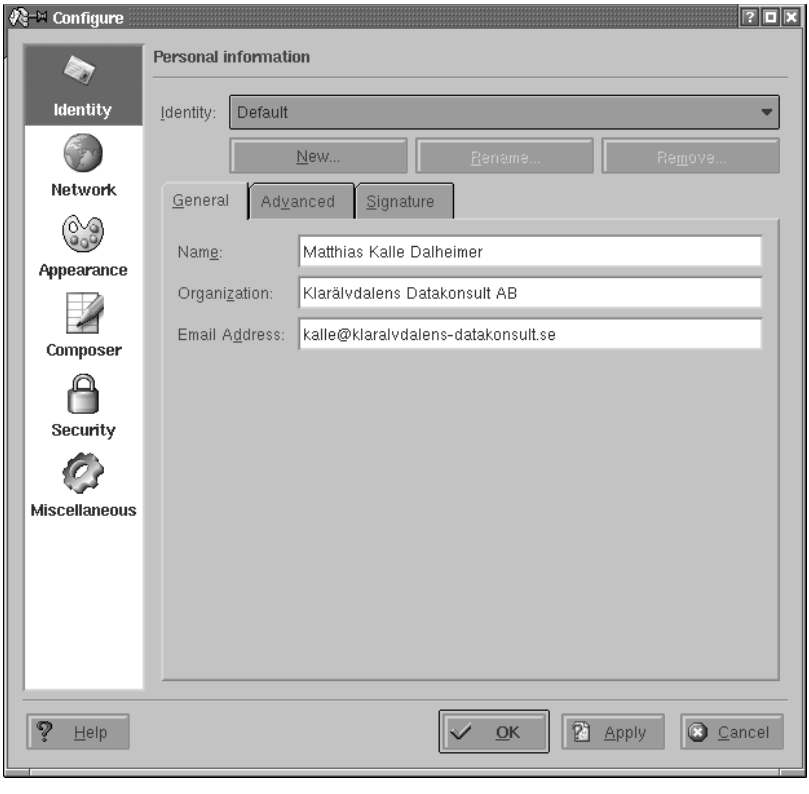


Рис. 6.2. *Настройка профиля в KMail*

Прежде чем начать работать с KMail, необходимо сообщить программе некоторую информацию. Выберите пункт Configure KMail (Настроить KMail) в меню Settings (Настройка), после чего откройте группу настроек Identity (Профили), щелкнув на ее значке. Здесь можно определить различные параметры, например, можно определить обратный адрес в зависимости от того, пишете ли вы письмо как служащий компании или как частное лицо. Щелкните на кнопке Add (Добавить), чтобы создать новый профиль. Появившийся диалог предложит вам на выбор три варианта создания нового профиля: с пустыми полями, с параметрами из центра управления KDE (этот вариант имеет смысл в том случае, если параметры электронной почты были ранее настроены в центре управления) и скопировать настройки из другого профиля (разумеется, такое возможно только при наличии хотя бы одного профиля и имеет смысл, если затем вы собираетесь отредактировать вновь созданный профиль). Обычно при первичной настройке KMail создается новый пустой профиль. Дайте профилю название, например «Рабочий» или «Домашний», и щелкните на кнопке OK. Для начала достаточно заполнить поля Name (Ваше имя) и Email Address (Электронный адрес) на вкладке General (Общие) (рис. 6.2).

Затем перейдите в группу настроек Accounts (Учетные записи). Здесь нужно создать хотя бы одну учетную запись для исходящей почты и одну – для входящей.

Начнем с исходящей почты, которая настраивается на вкладке Sending (Отправка) (рис. 6.3). Щелкните на кнопке Add (Добавить). Последует вопрос, будете ли вы использовать SMTP или работать непосредственно с установленной программой *Sendmail*. В большинстве случаев, когда имеется локально установленный MTA, требуется выбрать SMTP. Затем на вкладке General (Общие) диалога настройки транспорта SMTP задайте имя транспорта (его можно выбрать произвольно, поскольку оно нужно только вам, чтобы в дальнейшем различать настройки, а в сетевых взаимодействиях оно не участвует). В любом случае нужно указать имя сетевого узла и порт. Порт почти всегда имеет номер 25, а имя сетевого узла вам должен сообщить поставщик услуги. Если установлен локальный MTA и вы хотите работать с ним, просто укажите имя `localhost`. Если ваш почтовый сервер требует аутентификации (если не уверены, узнайте об этом у провайдера), установите соответствующий флажок и введите регистрационное имя и пароль. Перечисленных настроек должно хватить для отправки исходящей почты. Это самое малое, что необходимо сделать, однако большинство поставщиков услуг доступа в Интернет стараются защищаться от несанкционированного использования и практикуют обмен электронной почтой без аутентификации только с теми узлами Сети, которые имеют адреса IP, предоставляемые самим провайдером, либо требуют аутентификации, прежде чем принять почту от клиента, а затем отправляют эту почту по назначению.

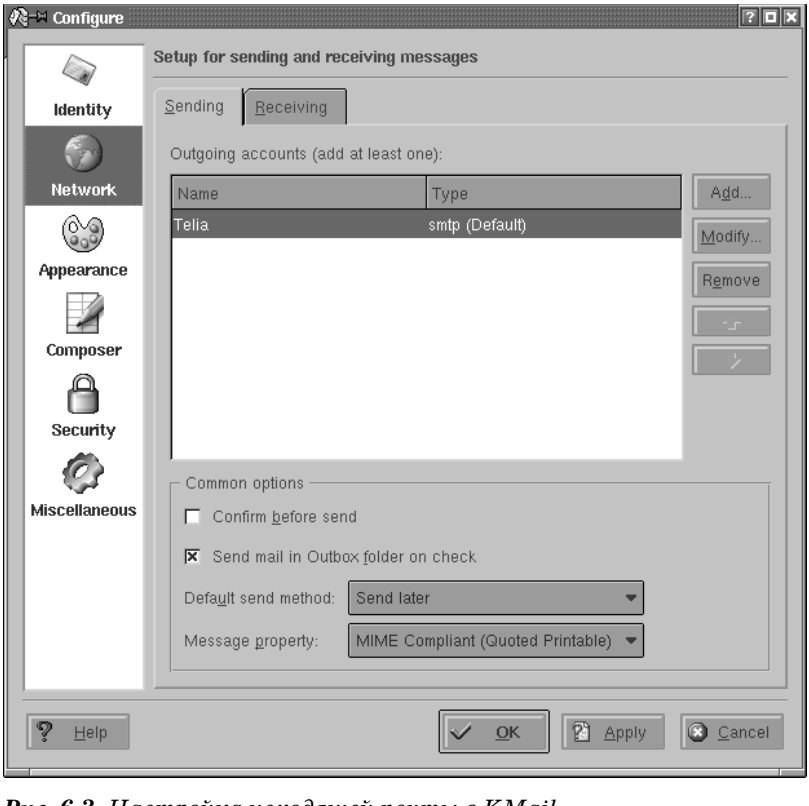


Рис. 6.3. *Настройка исходящей почты в KMail*

Этих настроек должно хватить, чтобы появилась возможность отправлять электронную почту, но мы рекомендуем выполнить некоторые дополнительные действия, чтобы установить максимальную степень защиты. KMail облегчает эту задачу, поскольку автоматически определяет настройки системы безопасности используемого сервера SMTP. Перейдите на вкладку Security (Безопасность) и щелкните на кнопке с надписью Check what the server supports (Проверить возможности сервера), чтобы узнать, какие настройки поддерживает сервер. KMail проверит соединение с сервером и выберет установки с наивысшей степенью защищенности. К сожалению, у многих провайдеров почтовые серверы работают вообще без всякой защиты.

Перейдем к настройке приемной части.[1] Закройте все диалоговые окна, чтобы вернуться к группе настройки Accounts (Учетные записи), и перейдите на вкладку Receiving (Получение). Здесь можно настроить несколько учетных записей, которые нужно опрашивать. Это удобно, если есть несколько провайдеров, которые хранят вашу электронную почту. Щелкните на кнопке Add (Добавить) и выберите тип почтового сервера. Если у вас работает свой локальный MTA, нужно выбрать параметр Local Mailbox (Локальный почтовый ящик). Обычно можно принять настройки по умолчанию, предлагаемые на следующей странице (при этом нужно изменить имя по умолчанию на что-либо более подходящее).

Если вы получаете сообщения непосредственно с сервера провайдера, нужно выбрать POP3 или IMAP[2], в зависимости от того, какой протокол поддерживает провайдер. В появившемся окне диалога снова введите выбранное вами имя, затем укажите имя регистрации, свой пароль, имя сетевого узла, на котором хранится ваша почта, и порт (обычно 110 для POP3 и 143 для IMAP). Всю эту информацию вам должен сообщить провайдер или системный администратор. Остальные параметры можно пока оставить как есть и поэкспериментировать с ними позднее.

Наиболее свежие версии KMail обладают возможностью извлекать почтовые сообщения, недоступные другим почтовым клиентам. Традиционно почтовый протокол IMAP требует прямого сетевого соединения с сервером, где хранится почта, потому что в соответствии с этим протоколом все сообщения сохраняются только на сервере. Однако KMail поддерживает режим под названием *разъединенный IMAP*, в котором все сообщения помещаются в локальный кэш, чтобы обеспечить возможность использовать достоинства протокола IMAP: например, содержимое ящика будет выглядеть одинаково с любого компьютера (с рабочей

---

[1] Это редко указывают явно, но вы можете настроить работу приемной (POP3 или IMAP4) и передающей (SMTP) частей одного почтового ящика для работы через совершенно разных провайдеров. Например, если у вас есть почтовые ящики у трех провайдеров, то все почтовые ящики вы настраиваете на получение почты от каждого из них соответственно, но отправку почты со всех трех ящиков настраиваете одинаково – через провайдера, скажем 2, потому что отправка от него сопровождается наименьшими задержками почты. – *Примеч. науч. ред.*

[2] Многие провайдеры предоставляют протоколы POP3 и IMAP4 на выбор: POP3 является более старым, более привычным и массово используемым протоколом, но многие рекомендации последнего времени сходятся на том, что использование менее привычного IMAP4 является более предпочтительной альтернативой. – *Примеч. науч. ред.*

станции или с ноутбука), и при этом можно работать автономно, когда в этом возникает потребность. Интеллектуальные механизмы синхронизации обеспечат одинаковое представление содержимого почтового ящика на всех компьютерах, с которых была выполнена попытка прочитать почту (разумеется, только после выполнения процедуры синхронизации).

Закройте все диалоги кнопкой OK. Теперь все готово для получения почты. Чтобы принять почту с сервера, выберите пункт меню File (Файл)→Check Mail (Проверить почту). В результате должны быть загружены все сообщения из всех заданных ящиков входящей почты, которые были настроены. Если этого не произойдет или будут получены сообщения об ошибках, проверьте еще раз все значения, которые вы ввели на разных страницах настройки, и сравните их с информацией, полученной от вашего провайдера или системного администратора. Обычно ошибки обусловлены опечатками в имени хоста, в имени пользователя или в пароле.

Если используется разъединенный IMAP, пункт меню Check Mail (Проверить почту) не просто проверит наличие новых сообщений в почтовом ящике на сервере, но и выполнит синхронизацию локального почтового ящика и ящика на сервере, включая удаление некоторых сообщений с сервера, изменение признаков и т. д.

Чтобы отправить сообщение, нажмите Ctrl+N или выберите пункт меню Message (Сообщение)→New Message (Новое сообщение). Откроется окно редактора сообщений, в котором можно ввести адрес получателя, тему и, собственно, тело сообщения. Функция автодополнения поможет своими подсказками в выборе адреса получателя при его наборе. Подсказки формируются на основе адресной книги (если она имеется) и почтовых адресов из входящих и исходящих отправлений.

Если вы установили несколько профилей, можно выбрать тот, который будет использован в данном сообщении. Завершив составление письма, нажмите Ctrl+N. В зависимости от настроек транспорта исходящей почты сообщение помещается в папку для исходящих сообщений (и будет ожидать там дальнейшей обработки – режим по умолчанию) или передается сразу же. Если вы хотите отменить эту настройку для конкретного сообщения, выберите в меню редактора сообщений пункт меню Message (Сообщение)→Queue (Отправить позже) или Message (Сообщение)→Send Now (Отправить сейчас).

Сообщения, помещенные в папку для исходящих, по умолчанию не отсылаются автоматически (можно, однако, настроить KMail так, чтобы исходящие сообщения всегда отправлялись при проверке входящей почты). Чтобы отправить все сообщения из папки для исходящих, выберите пункт File (Файл)→Send Queued (Отправить из очереди) в главном меню KMail. Мы взяли себе за правило никогда не отправлять сообщения автоматически и всегда просматривать содержимое папки для исходящих перед отправкой сообщений, что дает возможность избежать затруднений, которые могут возникнуть, когда письма будут отправлены не тем адресатам. Просмотр писем, выдержанных в резком тоне, которые были написаны в порыве чувств, после того, как гнев остыл, поможет сохранить дружеские отношения с вашими товарищами и деловыми партнерами.

Если возникнут проблемы при отправке сообщений, снова проверьте, не были ли допущены опечатки в настройках. Кроме того, для предотвращения пересылки нежелательной коммерческой почты (так называемого *спама*) через свои серверы некоторые провайдеры требуют, чтобы вы проверили свой почтовый ящик на

сервере (с предоставлением своего имени пользователя и пароля) и таким образом идентифицировали себя, прежде чем вам будет разрешено отправить электронную почту через этот сервер. После проверки входящей почты вам дается некоторое время (обычно 15 минут) на отправку исходящей почты.

Теперь у вас достаточно знаний о KMail, чтобы продолжить его освоение самостоятельно. Возможно, первое, что вам понадобится (особенно если вы каждый день получаете много почты!), – это создать новые папки, выбрав пункт меню Folder (Папка)→Create (Новая папка), и настроить фильтры с помощью пункта Settings (Настройки)→Configure Filters (Настроить фильтры). В результате вы сможете направлять сообщения с некоторыми характеристиками (скажем, с определенными адресатами или темами) в отдельные папки. Например, можно помещать все сообщения из почтового списка рассылки в специально выделенную папку. Если все, что вам нужно, – это лишь подшить сообщения, посланные по определенному списку рассылки, определенному получателю или с определенной темой, можно просто щелкнуть правой кнопкой мыши на заголовке нужного сообщения и выбрать в контекстном меню пункт Create Filter (Создать фильтр). Дополнительное подменю, которое появится при выборе этого пункта, поможет уточнить выбор. После этого появится диалог настройки фильтра с уже заполненными полями критерия, вам останется лишь определить, что делать с этими сообщениями – перемещать их в определенную папку или сразу же удалять.

Если со временем обнаружится, что вы регулярно пользуетесь не только программой KMail, но также адресной книгой Kaddressbook и календарем KOrganizer, входящими в состав KDE, и были бы непрочь интегрировать эти приложения в одно общее окно, тогда обратите внимание на программу Kontact. Это приложение-обертка, которое при помощи технологии Kparts объединяет отдельные программные компоненты в общий интерфейс, как показано на рис. 6.4.

Доступ к отдельным компонентам производится с помощью кнопок в боковой панели, расположенной с левой стороны окна. Чтобы запустить тот или иной компонент, необходимо щелкнуть на соответствующей ему кнопке. Для большинства приложений, входящих в Kontact, эти кнопки могут выступать в качестве конечной точки операций перетаскивания мышью, это означает, что можно, например, перетащить сообщение электронной почты на кнопку Todo (Задачи), чтобы создать новую задачу, связанную с этим сообщением. Попробуйте перетащить какие-нибудь элементы на кнопки и посмотрите, что из этого получится.

Следует отметить, что все компоненты, находящиеся внутри Kontact, – это те же самые приложения, которые могут работать и автономно, например KMail или KAddressbook. Это означает, что данные приложения можно запускать отдельно, когда по каким-либо причинам запускать Kontact было бы нежелательно, и продолжать работать с теми же самыми данными и настройками. Все функциональные возможности, доступные в Kontact, также доступны и в автономных приложениях. Поскольку Kontact использует KParts, он может быть расширен и другими компонентами, а не только теми, что поставляются вместе с ним; уже существуют компоненты сторонних производителей, например агрегаторы чтения лент новостей. Чтобы увидеть список установленных и имеющихся компонентов, выберите пункт меню Settings (Настройки)→Select Components (Выбор компонентов).

Одна из самых интересных возможностей, которой обладает Kontact, – это возможность представления сводной информации. Чтобы получить ее, нужно

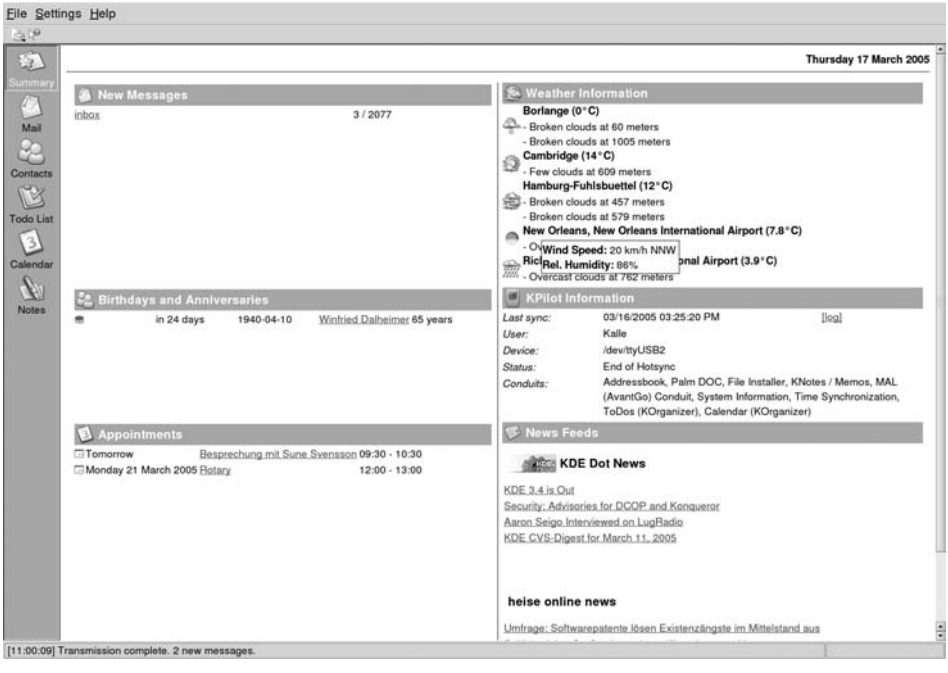


Рис. 6.4. *Общий вид окна программы Kontact*

щелкнуть на кнопке Summary (Дайджест) в боковой панели. Страница, которая появится вслед за этим, будет заполнена сведениями, поставляемыми каждым из встроенных компонентов. Так, в области отображения электронной почты будет выведено содержимое папок с непрочитанной почтой. Щелчок на одной из них перенесет вас непосредственно в эту папку. Точно так же компонент календаря покажет все приближающиеся события, дни рождения из записной книжки и еще не завершенные задачи. Чтобы настроить представление сводной информации, выберите пункт Configure Summary View (Настроить дайджест) в меню Settings (Настройки). В последних версиях Kontact имеется возможность переупорядочить отдельные области представления, просто перетаскивая их мышью за область заголовка.

Почтовый клиент Mozilla Mail & News

Mozilla Mail & News – это почтовый клиент, устанавливаемый вместе с веб-броузером Mozilla, если вы не выбрали минимальную конфигурацию, включающую только броузер и редактор сообщений. Вполне возможно, что в вашем дистрибутиве уже есть Mozilla. Если нет или вы хотите установить более новую версию, можно загрузить его с сайта *http://www.mozilla.org*. Компонент Mozilla Mail & News имеет самостоятельную версию – Thunderbird, которую можно найти по адресу *http://www.mozilla.org/products/thunderbird*.[1] Этот почтовый клиент пре-

[1] Русскоязычный ресурс: *http://www.mozilla-russia.org/*. – *Примеч. науч. ред.*

красно справляется с фильтрацией спама и защищает систему от проникновения почтовых вирусов.

Основные идеи установки и работы Mozilla Mail почти такие же, как для KMail, поэтому мы покажем только основные различия. Чтобы открыть почтовый клиент, запустите Mozilla и выберите в меню пункт Windows (Окно)→Mail and Newsgroups (Почта и новости). При первом запуске почтовой программы появится мастер настройки электронной почты. Установите учетную запись электронной почты на первой странице и свои идентификационные данные на второй (работа Mozilla с учетными записями не столь гибкая, как в KMail, поскольку идентификационные данные здесь привязаны к учетным записям, тогда как в KMail идентификационные данные можно менять произвольно).

На третьей странице выберите POP или IMAP в качестве протокола для получения входящей почты (локально получать почту с помощью Mozilla Mail & News нельзя, что является большим недостатком) и укажите имена серверов входящей и исходящей почты (укажите *localhost* в обоих случаях, если вы работаете с собственным MTA). Введите остальные сведения на следующих страницах, и вы готовы к запуску Mozilla Mail & News. Размещение информации на экране по умолчанию в точности такое же, как в KMail.

Как и при работе в KMail, первое, что вам, возможно, потребуется в Mozilla Mail & News, – это организовать дополнительные папки и фильтры для сортировки входящих сообщений. Создать новую папку можно, щелкнув правой кнопкой на списке папок и выбрав в появившемся контекстном меню пункт New Folder (Создать папку). Правила для фильтров можно настроить с помощью меню Tools (Сервис)→Message Filters (Фильтры сообщений).

На этом наше обсуждение использования клиентов электронной почты в Linux завершается. Как вы видели, существует много параметров, от самых простых до сложных, с помощью которых можно администрировать и обрабатывать ежедневный поток сообщений электронной почты.

Получение почты с помощью Fetchmail

Если провайдер хранит вашу почту, пока вы ее не заберете, и вы не хотите использовать свою почтовую программу для загрузки почты, вам потребуется программа, которая забирает почту с компьютера провайдера. Существует множество таких программ. Мы коротко расскажем о *fetchmail*, так как она устойчива и гибка и может работать и с POP3, и с IMAP.

Можно загрузить *fetchmail* из своего любимого архива Linux, и, скорее всего, она также входит в состав вашего дистрибутива. Если вы скачали исходный код *fetchmail*, распакуйте, соберите и установите программу в соответствии с инструкциями. Текущей версией на момент написания книги была 6.2.5. Адрес официальной домашней страницы *fetchmail*: *http://www.catb.org/~esr/fetchmail/*.

Поведением *fetchmail* можно управлять через параметры командной строки или через файл с настройками. Неплохо сначала попробовать забрать свою почту, передавая всю необходимую информацию через параметры командной строки, и в случае успеха написать файл конфигурации.

Для примера предположим, что провайдер использует протокол POP3, имя пользователя – *joeuser*, а его пароль – `secret`. Имя машины, на которой работает POP3-сервер, – *mail.isp.com*. Тогда пользователь может забрать свою почту с помощью следующей команды:

```
fetchmail --protocol POP3 --username joeuser mail.isp.com
```

Затем *fetchmail* запросит пароль и после того, как пользователь введет его правильно, заберет почту и передаст ее MTA для дальнейшей доставки. Здесь мы предположили, что SMTP-сервер работает на локальной машине и для приема запросов использует порт 25. Так и должно быть, если MTA был установлен корректно.

Экспериментируя с *fetchmail*, нелишним будет указать параметр `--keep`. В этом случае *fetchmail* не удалит почту из вашего POP3-ящика. Обычно все сообщения удаляются с жесткого диска провайдера после того, как они доставлены на локальную машину пользователя. Большинство провайдеров ограничивают объем почтовых отправлений, который можно хранить на их машинах, и если вы не будете удалять почту с сервера провайдера, можно дойти до пределов установленных квот. Однако на время тестирования стоит использовать `--keep`, чтобы не потерять ни одного письма.

Используя рассмотренные параметры *fetchmail*, вы сможете в большинстве случаев забрать свою почту. Например, если провайдер использует новый протокол IMAP, просто укажите в командной строке `IMAP` вместо `POP3`. Если провайдер использует необычные установки, вам может понадобиться один из параметров, о которых вы узнаете на страницах справочного руководства для *fetchmail*(1).

Если вы удовлетворены процессом загрузки, можно сохранить все необходимые настройки *fetchmail* в файле, чтобы не вводить все параметры вручную. Этот файл называется *.fetchmailrc* и должен храниться в исходном каталоге. Закончив его редактирование, убедитесь, что он имеет значение разрешений, равное 0600, чтобы никто кроме вас не мог прочитать его, так как в нем может храниться ваш пароль:

```
chmod 0600 ~/.fetchmailrc
```

Полный синтаксис файла с настройками описан на страницах справочного руководства по *fetchmail*, но обычно нужны только очень простые строки, начинающиеся с `poll`. Чтобы определить те же параметры, что и в командной строке предыдущего примера, но на этот раз вместе с паролем, надо записать в файл с настройками следующую строку:

```
poll mail.isp.com protocol pop3 username joeuser password secret
```

Теперь можно запускать *fetchmail* без параметров. На этот раз *fetchmail* не будет запрашивать пароль, поскольку уже знает его из файла с настройками. Для безопасности можно добавить слово `keep` в строку `poll`.

Использование *fetchmail* с файлом конфигурации дает еще одно преимущество: можно забирать почту из любого количества почтовых ящиков. Для этого нужно просто добавить другие строки `poll` в файл *.fetchmailrc*, и *fetchmail* по очереди заберет почту с разных серверов.

Как и когда запускать *fetchmail*, зависит от способа соединения с Интернетом. При постоянном подключении или неограниченном доступе можно вызывать

fetchmail с помощью *cron* с приемлемым интервалом (например, один раз в час). Но при непостоянном и дорогостоящем подключении к Интернету стоит запускать *fetchmail* вручную, когда вам действительно нужно получить и прочесть почту, чтобы минимизировать время соединения. Наконец, используя PPP для доступа к провайдеру Интернета, можно добавить вызов *fetchmail* из сценария `ip-up`, который запускается сразу после установки соединения с Интернетом. При такой настройке почта будет автоматически забираться, когда вы просматриваете веб-страницу и ваш компьютер дозванивается до провайдера.

А что происходит с почтовыми сообщениями после того, как *fetchmail* заберет их с почтового сервера? Ранее уже говорилось, что они будут переданы вашему MTA. После этого MTA, как правило, перемещает сообщения в так называемый локальный файл спулинга, который обычно называется */var/spool/mail/<имя пользователя>*. Затем следует настроить программу почтового клиента, чтобы она забирала сообщения из этого файла. В каждом клиенте должна существовать такая настройка, например, в KMail можно создать локальную учетную запись для получения.

Шифрование с помощью GnuPG

С помощью GNU Privacy Guard, или сокращенно GnuPG, можно выполнять шифрование файлов и сообщений электронной почты и добавлять к ним цифровую подпись. Основной инструмент командной строки GnuPG – *gpg*, он получил такое название потому, что начинался как замена PGP, который был первым инструментом шифрования, доступным каждому. Название PGP происходит от Pretty Good Privacy. Данный инструмент был написан Филом Циммерманном (Phil Zimmermann) в начале 90-х годов. OpenPGP – это стандарт, описывающий формат файла PGP версии 5.0 или выше. И GnuPG, и PGP следуют этому стандарту и поэтому могут читать файлы друг друга.

Симметричное шифрование

Самый простой способ зашифровать файл с помощью GnuPG состоит в том, чтобы зашифровать его ключевой фразой.[1] Такой метод называют *симметричным шифрованием*. Рассмотрение основ криптографии выходит далеко за рамки этой книги, поэтому мы скажем лишь, что ключевая фраза используется в качестве ключа шифрования файла. Каждый, кому известна ключевая фраза, будет в состоянии расшифровать и прочитать файл.[2]

Чтобы зашифровать файл с именем *music.ogg*, достаточно дать команду `gpg -symmetric music.ogg`. После этого GnuPG попросит дважды ввести ключевую фразу, чтобы избежать опечаток. Зашифрованный файл будет сохранен на диске под

[1] *Ключевая фраза* – это просто длинный пароль, состоящий из нескольких слов; обычно какое-нибудь предложение.

[2] Разумеется, шифровать можно не только текстовые файлы, но и файлы любого другого типа, поэтому не стоит понимать наши слова «прочитать файл» слишком буквально; здесь точно так же можно было бы написать «прослушать аудиофайл» или «просмотреть видеофайл».

именем *music.ogg.gpg*. Если необходимо дать файлу другое имя, следует использовать параметр командной строки *--output outfile*, например:

```
gpg --output music.gpg -c music.ogg
```

Здесь использованы параметры *-c* и *-o*, которые являются сокращенными вариантами параметров *--symmetric* и *--output* соответственно.

Чтобы расшифровать файл, нужно вызвать команду `gpg file`. Например, в продолжение предыдущего примера:

```
gpg music.ogg.gpg
```

Как и в предыдущем случае, с помощью параметра `-o outfile` можно определить имя расшифрованного файла.

Криптография на основе открытого ключа

Симметричное шифрование неплохо подходит для кратковременного использования или от случая к случаю, но вы неизбежно столкнетесь с проблемой запоминания ключевых фраз, накопившихся в изобилии после шифрования большого числа файлов. Очевидное решение, заключающееся в многократном использовании одной и той же ключевой фразы, породит проблемы, аналогичные тем, как если бы все ваши двери отпирались одним ключом. Кроме того, после потери одного ключа появляется вероятность, что недобропорядочный гражданин найдет его и обчистит ваше жилище. Более кратко эта проблема может быть описана так: «Любой, кто знает ключевую фразу, сможет прочитать содержимое файла».

Еще одна проблема кроется в том, что «каждый, кто должен иметь возможность прочитать содержимое файла, должен знать ключевую фразу». Если файл был зашифрован не для того, чтобы положить его в архив, а чтобы передать его своим друзьям, коллегам или деловыми партнерам, вы неизбежно столкнетесь и с этой проблемой. Нельзя многократно использовать одну и ту же ключевую фразу и пребывать в полной безопасности, потому что каждый новый файл мог передаваться другой группе получателей. Например, допустим, что некая ключевая фраза использовалась для шифрования сообщения Алисе и Бобу, а затем для шифрования другого сообщения, на сей раз Алисе и Чарли. Теперь Алиса, Боб и Чарли смогут прочитать оба сообщения, хотя изначально предполагалось, что только Алиса должна была иметь возможность прочитать оба сообщения.

Невозможно шифровать каждое сообщение новой ключевой фразой, поскольку получателю она будет неизвестна. А если у вас есть надежный канал передачи новой ключевой фразы, тогда вообще зачем вам шифрование?

Единственное решение при использовании простого шифрования состоит в том, чтобы договориться о ключевой фразе с каждым получателем в отдельности и шифровать сообщение каждому из них отдельно. Но и такой вариант порождает свои сложности, потому для каждой пары людей должна существовать уникальная ключевая фраза; проблема, как говорят, имеет сложность $O(n^2)$.

Эти проблемы не давали покоя специалистам в области криптографии до середины 70-х годов, когда Уайтфилд Диффи (Whitefield Diffie) и Мартин Хеллман (Martin Hellman) изобрели новый метод обмена ключами, который не требовал передачи секретного ключа другим лицам. Они использовали асимметричное шифрование, когда ключ шифрования общеизвестен, но ключ к расшифровке

хранится в секрете. При использовании этой схемы любой желающий сможет зашифровать письмо, например, для Алисы, но только Алиса сможет расшифровать его своим секретным ключом.

Этот метод существенно упрощает ситуации, описанные выше: сообщения каждому получателю шифруются с помощью их открытых ключей, и только сами получатели смогут прочитать зашифрованные сообщения. Кроме того, секретный ключ существует в единственном экземпляре – у его владельца, а не в двух; степень сложности проблемы уменьшилась до уровня O(n). Правда, здесь возникает другая проблема – проблема ошибочного шифрования сообщений, отправляемых одному человеку, открытым ключом другого человека, поскольку открытые ключи можно без труда загрузить с сервера ключей, но мы пока оставим ее и вернемся к ней в разделе «Сеть доверия».

Создание новой пары ключей

Чтобы иметь возможность передавать и принимать сообщения, зашифрованные открытым ключом, необходимо иметь пару ключей – открытый и закрытый. Они могут быть созданы с помощью команды `gpg --gen-key`. По этой команде GnuPG задаст серию вопросов, после чего сгенерирует новую пару ключей. Ниже приводится пример прохождения процедуры создания ключей в GnuPG версии 1.4.0. В самом конце GnuPG просит ввести фразу, которая будет служить защитой для закрытого ключа. Этот ключ *не* может использоваться для шифрования.

```
$ gpg --gen-key
gpg (GnuPG) 1.4.0; Copyright (C) 2004 Free Software Foundation, Inc.
This program comes with ABSOLUTELY NO WARRANTY.
This is free software, and you are welcome to redistribute it
under certain conditions. See the file COPYING for details.

gpg: создан каталог `/home/john/.gnupg'
gpg: создан новый файл настроек `/home/john/.gnupg/gpg.conf'
gpg: ВНИМАНИЕ: параметры в `/home/john/.gnupg/gpg.conf' еще неактивны при этом запуске
gpg: создана таблица ключей `/home/john/.gnupg/secring.gpg'
gpg: создана таблица ключей `/home/john/.gnupg/pubring.gpg'
Выберите тип ключа:
   (1) DSA и ElGamal (по умолчанию)
   (2) DSA (только для подписи)
   (5) RSA (только для подписи)
Ваш выбор (?-подробнее)? 1
Пара ключей DSA будет иметь длину 1024 бит.
ключи ELG-E могут иметь длину от 1024 до 4096 бит.
Какой размер ключа Вам необходим? (2048) 2048
Запрашиваемый размер ключа 2048 бит
Выберите срок действия ключа.
         0 = без ограничения срока действительности
      <n>  = срок действительности n дней
      <n>w = срок действительности n недель
      <n>m = срок действительности n месяцев
      <n>y = срок действительности n лет
Ключ действителен до? (0) 5y
Ключ действителен до: Чтв 12 Янв 2012 07:45:59 MSK
Все верно? (y/N) y
```

```
Для идентификации Вашего ключа необходим User ID
Программа создаст его из Вашего имени, комментария и адреса e-mail в виде:
    "Heinrich Heine (Der Dichter) <heinrichh@duesseldorf.de>"

Ваше настоящее имя: John Doe
Email-адрес: john@doe.example.net
Комментарий: work
Вы выбрали следующий User ID:
    "John Doe (work) <john@doe.example.net>"

Сменить (N)Имя, (C)Комментарий, (E)email-адрес или (O)Принять/(Q)Выход? O
Для защиты секретного ключа необходим пароль.
Введите пароль:
Повторите пароль:
Необходимо сгенерировать много случайных чисел. Желательно, чтобы Вы
выполняли некоторые другие действия (печать на клавиатуре, движения мышью,
обращения к дискам) в процессе генерации; это даст генератору
случайных чисел возможность получить лучшую энтропию.
.++++++++++.+++++.++++++++++.+++++.+++++++++++++++.+++++..+++++++++++++++++++++++++++++
+.....++++++++++..+++++..+++++...+++++++++++++++.....++++++++++++++++++++..+++++

Недостаточно случайных чисел. Выполняйте какие-либо действия для того,
чтобы ОС могла получить больше случайных данных! (Необходимо еще 284 байт)

Необходимо сгенерировать много случайных чисел. Желательно, чтобы Вы
выполняли некоторые другие действия (печать на клавиатуре, движения мышью,
обращения к дискам) в процессе генерации; это даст генератору
случайных чисел возможность получить лучшую энтропию.
.....+++++.++++++++++++++.+++++.++++++++++++++++++++++++++++++++.++++++++++.+++++++
++++++++..++++++++++++++++++.+++++++++++++++..+++++++++++++++++++++++++>.++++++++++>.
.+++++>.+++++.........>+++++..<.+++++.............................>+++++<+++++.......<++
+++...>....+++++.........<+++++.........................................+++++^^^
gpg: /home/john/.gnupg/trustdb.gpg: создана таблица доверий
gpg: ключ 79AF7161 помечен как абсолютно доверяемый.
открытый и закрытый ключи созданы и подписаны.

gpg: проверка таблицы доверий
gpg: 3 ограниченных необходимо, 1 выполненных необходимо, PGP модель доверия
gpg: глубина: 0  корректных:   1  подписанных:   0  доверия: 0-, 0q, 0n, 0m, 0f, 1u
gpg: срок следующей проверки таблицы доверий 2012-01-12
pub   1024D/79AF7161 2007-01-13 [годен до: 2012-01-12]
Отпечаток ключа = 4C3F AF05 CAAC 7835 4336  F90B CC81 F624 79AF 7161
uid                  John Doe (work) <john@doe.example.net>
sub   2048g/D8033544 2007-01-13 [годен до: 2012-01-12]
```

После создания пары ключей GnuPG сохранит их в локальной «связке ключей» — в каталоге *~/.gnupg*. Убедиться в благополучном создании ключей можно с помощью команды `gpg --list-keys`, которая выведет список ключей из связки открытых ключей, и `gpg --list-secret-keys` — из связки закрытых.

Чтобы сделать открытый ключ доступным для тех, кто пожелает отправлять вам зашифрованные послания, его следует выгрузить на сервер ключей с помощью команды:

```
gpg --keyserver wwwkeys.pgp.net --send key-id
```

где *key-id* — это идентификатор ключа (в данном примере 79AF7161). Имя сервера ключей можно определить в файле настроек *~/.gnupg/gpg.conf*, чтобы избежать необходимости ввода его имени в командной строке всякий раз, когда потребуется выгрузить или загрузить ключи. Нет никакой необходимости выгружать открытый ключ более чем на один сервер ключей, поскольку серверы *pgp.net* сами обмениваются между собой вновь поступившими ключами.

Теперь самое время обсудить проблему утраты ключевой фразы: если ключ был скомпрометирован или вы просто забыли ключевую фразу, необходимо известить всех, кто использовал ваш открытый ключ, что он больше не действителен. На этот случай существует *сертификат отзыва*. Отозванный ключ не может больше использоваться для шифрования. Однако, чтобы создать сертификат отзыва, необходимо знать ключевую фразу, которая отпирает закрытый ключ. Таким образом, сертификат отзыва следует подготовить заранее, чтобы издать его, как только сложится критическая ситуация, а поэтому его необходимо создать еще до того, как ключевая фраза будет забыта, и сохранить где-нибудь в закрытом от посторонних месте.

Создать сертификат отзыва можно командой *gpg --armour --output rev-cert.gpg --gen-revoke key-id*. Эта команда создаст сертификат отзыва и сохранит его в файле *rev-cert.gpg*. Параметр `--armour` предупреждает GnuPG, что следует создать печатаемую версию сертификата, а не двоичный файл. Благодаря этому можно будет распечатать сертификат на бумаге и спрятать в сейф на случай потери информации с жесткого диска.

Чтобы воспользоваться сертификатом отзыва, достаточно импортировать его командой `gpg < rev-cert.gpg` и выгрузить измененный ключ командой `gpg --send` *key-id*, как это было показано ранее.

Ключи, выгруженные на сервер ключей, не могут быть удалены. Кроме того, они могут только добавляться туда, никакая информация не может быть удалена с серверов, включая дополнительные идентификаторы пользователей, цифровые подписи сторонних производителей (вскоре эта тема будет рассмотрена) и уже отозванные ключи.

Шифрование открытым ключом

Как говорилось ранее, чтобы зашифровать послание открытым ключом получателя, этот ключ сначала необходимо получить. В терминах GnuPG это означает, что ключ необходимо загрузить с сервера ключей, и что путь от ключа получателя до ключа, который загрузите вы, должен быть абсолютно надежным (об этом рассказывается в разделе «Сеть доверия» далее в этой главе).

Пока мы можем воспользоваться одной из особенностей GnuPG: шифрование ненадежными ключами.

Но для начала необходимо отыскать ключ на сервере ключей. Для этого можно воспользоваться возможностями GnuPG: `gpg --search <`*имя-или-адрес-электронной-почты*`>`. В результате выполнения команды GnuPG представит список всех ключей, совпадающих с критерием поиска (а их могут быть сотни), из которого можно будет отобрать требуемый.

Если известен идентификатор искомого ключа, можно сразу же загрузить ключ командой `gpg --recv key-id`.

Далее можно выполнить шифрование файла с использованием одного или более ключей. GnuPG позволяет выполнять шифрование не только вашим, но и любым другим открытым ключом (можно определить в файле с настройками), поэтому вы уже не сможете расшифровать зашифрованное сообщение. Выполняется шифрование командой:

```
gpg --encrypt --recipient recip_1 --recipient recip_2 ... file
```

При использовании коротких версий параметров:

```
gpg -e -r recip_1 -r recip_2 ... file
```

Обе команды создадут зашифрованное сообщение в файле с именем *file.gpg*, которое можно изменить с помощью параметра перенаправления вывода в файл с другим именем *--output* (*-o*). Не важно, сколько открытых ключей использовалось в процессе шифрования, всегда будет получаться только один файл, просто он будет создан таким образом, что каждый из указанных получателей сможет расшифровать его.

Чтобы расшифровать файл, достаточно просто передать его GnuPG: *gpg file.gpg*. В этом случае GnuPG попросит ввести ключевую фразу и сохранит расшифрованный файл под именем *file* (то есть из имени исходного файла будет убрано расширение *.gpg)*.

Если необходимо зашифровать сразу несколько файлов, следует воспользоваться параметром командной строки *--multifile*, например:

```
gpg --multifile -e -r recip_1 ... file1 file2 ...
```

Цифровые подписи

Криптография на основе открытого ключа может использоваться не только для шифрования, но и для аутентификации. Цифровые подписи дают гарантию, что файл не был изменен с момента его подписания. Проще говоря, система шифрования добавляет контрольную сумму с помощью закрытого ключа. Такой способ возможен благодаря тому, что получатель может открытым ключом расшифровать данные, зашифрованные закрытым ключом.

Таким образом, чтобы убедиться в подлинности цифровой подписи, получатель тоже вычисляет контрольную сумму и сравнивает ее значение с тем, что было сохранено в подписи. Если они совпадут, можно считать доказанным следующее: во-первых, данные не были изменены, так как они были подписаны, и во-вторых, сообщение было подписано с помощью закрытого ключа. Если данные были изменены, контрольные суммы совпадать не будут. Аналогично, если оригинальная контрольная сумма была зашифрована некоторым другим закрытым ключом, результат расшифровки с помощью открытого ключа будет представлять собой бессмысленный набор символов и контрольные суммы также не будут совпадать.

OpenPGP предоставляет два способа добавления цифровых подписей: встраивание и отделение. Встроенная цифровая подпись добавляется в исходное сообщение, то есть сами данные и цифровая подпись сохраняются в одном файле. В случае с отдельной цифровой подписью исходный файл не изменяется, а подпись со-

храняется во втором файле, в этом случае обычно получается два файла с расширениями *.gpg* и *.asc*. Желательно использовать метод с отдельной подписью, потому что он может использоваться для подписания файлов любого типа, тогда как метод со встроенной подписью – только для простых текстовых файлов.

Чтобы подписать файл, необходим только ваш закрытый ключ. Чтобы создать цифровую подпись к файлу *music.ogg*, надо выполнить следующую команду:

```
gpg --sign music.ogg
```

Подпись будет сохранена в файле *music.ogg.gpg*. Как обычно, имя файла с подписью по умолчанию можно переопределить с помощью параметра командной строки *--output* (*-o*). Сокращенная версия параметра *--sign* – *-s*.

Для вас уже не должно быть сюрпризом, что проверка подлинности цифровой подписи выполняется просто передачей файла подписи системе GnuPG: `gpg music.ogg.gpg`.

Операции шифрования и добавления цифровой подписи можно объединять в одной команде. В действительности эта самая обычная операция:

```
gpg -es -r recip_1 -r recip_2 ... file
```

Обратите внимание: в данном случае подпись шифруется вместе с подписываемыми данными, таким образом, вы не увидите третий файл с подписью. Все это будет упаковано в файл *.gpg*.

Обратите внимание: к моменту написания этих строк возможность создания подписей с параметром *--multifile* еще не была реализована. Чтобы подписать несколько файлов, придется воспользоваться оператором цикла `for` языка командной оболочки:

```
for i in *.ogg; do gpg --sign $i ; done
```

Сеть доверия

Выше уже отмечалось, что для шифрования с открытым ключом необходима уверенность в подлинности ключа, полученного от сервера ключей, то есть в том, что ключ не был изменен или подделан.

В настоящий момент в OpenPGP используется понятие «Сеть доверия», когда подлинность одних открытых ключей может удостоверяться цифровыми подписями других людей. Делается это путем подписания открытого ключа и идентификатора пользователя.

Рассмотрим такой пример: Алиса хочет послать зашифрованное письмо Бобу, которое не смогла бы прочитать ее подруга Кэрол. Она не знакома с Бобом лично и не может быть уверена в том, что открытый ключ, который она нашла на сервере по имени «Боб», действительно принадлежит именно этому Бобу, а не какому-нибудь другому.

Однако она лично знакома с Кэрол, и в свое время они подписали открытые ключи друг друга. Это означает, что открытый ключ Кэрол содержит подпись Алисы, которая означает примерно следующее: «Я, Алиса, подтверждаю, что этот ключ действительно принадлежит его владельцу, имя которого, то есть Кэрол, приведено в идентификаторе пользователя».

Кэрол, в свою очередь, знакома с Бобом.[1] Во времена, когда они еще были вместе, они также подписали открытые ключи друг друга, чтобы иметь возможность писать друг другу любовные письма.

Если Алиса *доверяет* Кэрол (она не настолько безрассудна, чтобы подписывать ключи тех, кому она не доверяет), она может использовать ключ Кэрол для создания *пути доверия* (*trust path*) к Бобу: ее собственная подпись удостоверяет подлинность ключа Кэрол. Она *доверяет* владельцу ключа, который может подписать другие ключи, и сообщает об этом GnuPG, определив соответствующий *индекс доверия* (*ownertrust*) для ключа Кэрол. Поскольку ключ Боба несет в себе удостоверяющую подпись Кэрол, он будет считаться действительным с точки зрения Алисы.

Этот пример основан на двух фундаментальных положениях:

- Подлинность ключа не может быть абсолютной. Она всегда относительна и зависит от другого ключа и степени доверия других людей к владельцу ключа, что выражается в виде присвоенных индексов доверия. Если бы Алиса не доверяла Кэрол настолько, что решилась подписать ее ключ, она не смогла бы убедиться в подлинности ключа Боба.
- Модель Сети доверия прекрасно работает в узком кругу лиц. Однако эта модель практически не работает далее одного-двух шагов (то есть промежуточных ключей).

В последние годы, однако, глобальная Сеть доверия (global Web of Trust) переживала бурное развитие, поэтому последний пункт теряет свое значение. Благодаря инструментальным средствам анализа, которые создали и запускают каждые две недели Дрю М. Стриб (Drew M. Streib) и Джейсон Харрис (Jason Harris), теперь известно, что глобальная Сеть доверия содержит один большой *строго связанный* набор ключей, в котором можно построить путь доверия к любому ключу в наборе. Этот набор в настоящее время охватывает 28 418 ключей, а его диаметр составляет примерно 15 шагов. От любого ключа из этого набора на расстоянии не более 30 шагов находятся еще 60 000 ключей. Обычно на расстоянии 3 шагов находятся приблизительно 10 000 ключей. Среднеквадратичное расстояние между любыми ключами в кластере в настоящее время составляет 3,6 шага. В противоположность этому набору величина ближайших крупных наборов составляет 147, 117 и 79 ключей.

Чтобы войти в состав строго связанного набора, необходимо и достаточно подписать ключи друг друга с одним и членов этого набора. При проживании в Северной Америке или Европе это не вызывает больших проблем. Посещения конференций и встреч с разработчиками Debian или KDE позволят принять участие в одной из многих акций подписания ключей, происходящих на этих мероприятиях. Однако в других частях мира это может быть проблематично.

Операции подписания ключа и изменения индекса доверия выполняются командой `gpg --edit` *`key-id`*, которая открывает вход в командную оболочку GnuPG, где доступны такие команды управления ключами, как `sign` и `trust`.

[1] Хотя часто жалеет об этом.

Агент gPG

Поработав некоторое время с GnuPG, можно заметить, что ключевую фразу приходится вводить достаточно часто. Но это не должно служить причиной выбора короткой фразы! Лучше обратиться к услугам *gpg-agent*.

Как и *ssh-agent*, *gpg-agent* можно настроить с целью поддержки кэша последних введенных ключевых фраз и использовать этот кэш, чтобы не запрашивать ввод фраз у пользователя. *gpg-agent* входит в состав GnuPG 2 – GnuPG следующего поколения. Загрузить версию GnuPG 2 можно с сайта *ftp://ftp.gnupg.org/gcrypt/alpha/gnupg*; где файлы пакетов с ним носят имена *gnupg-1.9.n*. Даже несмотря на то, что *gpg-agent* предназначался для работы в составе GnuPG 2, он прекрасно справляется со своими обязанностями с GnuPG версии 1.2.6 и выше. Обратите внимание: *gpg-agent* использует пакет *pinentry*, с помощью которого запрашивает ввод ключевой фразы. В настоящее время существуют версии *pinentry* для Qt (KDE), GTK (GNOME) и ncurses (текстовый терминал).

Чтобы заставить GnuPG использовать агент, сначала нужно запустить его: `eval` `` `gpg-agent --daemon` ``. Оператор `eval` передает командной оболочке вывод, полученный от команды, в обратных одиночных кавычках. Это очень важно, потому что команда `gpg-agent` выводит строку, в которой производится присваивание значения переменой среды окружения, что в свою очередь необходимо для того, чтобы агент мог использоваться GnuPG в процессе работы. В данном случае речь идет о переменной окружения `GPG_AGENT_INFO`. Если GnuPG запускается из этой же оболочки (или любой другой, порожденной от этой) и ему передается параметр `--use-agent` (или в командной строке, или в файле с настройками *~/.gnupg/gpg.conf*), то GnuPG, вместо того чтобы запрашивать ключевую фразу у пользователя, будет обращаться за ней к *gpg-agent*.

Чтобы заставить *gpg-agent* кэшировать ключевые фразы, а не запрашивать их каждый раз, необходимо создать файл *~/.gnupg/gpg-agent.conf* со следующим содержимым:

```
default-cache-ttl
            3600
```

Этот параметр сообщает, что *gpg-agent* должен удерживать ключевые фразы в кэше 3600 секунд, то есть один час.

7

Игры

Операционная система Linux долгое время пользовалась недоброй репутацией у любителей компьютерных игр. Даже очень опытные пользователи зачастую оставляли на жестком диске раздел с Windows только ради возможности поиграть в игры. Эта проблема напоминает старую проблему курицы и яйца: разработчики игр не торопятся переносить их на платформу Linux, потому что пользователей у этой операционной системы не так много, а рост числа пользователей сдерживается (кроме всего прочего) недостатком игр.

Однако положение с играми под Linux с каждым годом продолжает улучшаться. Не только основные производители видеокарт стремятся обеспечить полную поддержку ускорения трехмерной графики под X, но и множество компаний, разработчиков программного обеспечения, такие как Id Software и Epic Games, постоянно выпускают версии своих игр для Linux. Конечно, в чем-то это обусловлено сильной стороной Linux как серверной платформы. Идея заключается в том, что чем больше пользователей Linux компании привлекут, тем более вероятно появление серверов Linux для игр.

Если посмотреть на перечень коммерческих игр, перенесенных в Linux, можно без труда заметить, что в основном это шутеры (shooter – «стрелялки»): Doom, полная серия Quake, серия Unreal Tournament, Return to Castle Wolfenstein, Tribes 2 и многие другие. Это вовсе не означает, что игры других жанров отсутствуют в Linux – например, игры Railroad Tycoon и NeverWinter Nights также были перенесены в Linux, вот только создается впечатление, что игры-стрелялки переносятся с куда большей готовностью.

Даже если ваша любимая игра не была перенесена в Linux, все равно есть шанс, что двоичные файлы игры в версии для Windows можно будет установить и запускать в окружении Wine или Cedega. Эти две среды преобразуют системные вызовы Windows в системные вызовы Linux, и многие игры работают в них вполне прилично. Cedega – это коммерческий продукт, выпущенный компанией Transgaming. Он основан на Wine, а основные усилия разработчиков были сконцентрированы на том, чтобы обеспечить работоспособность последних версий игр под Linux.

На сайте Cedega можно найти обширный список игр, поддерживаемых их платформой, с оценкой качества работы под Linux. В список включены такие игры, как Warcraft III, Max Payne II и Battlefield 1942. Если вы решите воспользоваться Cedega, можно подписаться на базовую рассылку всего за $5 в месяц на сайте *www.transgaming.com*. На сайте имеется большое количество документов с вопросами и ответами на них, касающимися поддержки различных игр и их установки.

Существует немало эмуляторов игровых приставок под Linux. Если в вашем распоряжении имеются образы картриджей от игровых приставок, можно воспользоваться соответствующим эмулятором, таким как Xmame, Nestra или Snes9x, и поиграть в эти игры, находясь в системе Linux. Некоторые любители имеют целые шкафы, заполненные большими коллекциями игр и специализированными джойстиками.

Но перечень игр под Linux не ограничивается одними коммерческими версиями. Для Linux существует достаточно много свободно распространяемых игр. Существуют самые разнообразные игры, начиная от простых карточных и настольных игр, таких как шахматы или нарды, и простых аркадных игр, как, например, xgalaga, до сложных игр приключенческого жанра, таких как rogue и nethack. Помимо этого среди свободно распространяемых игр существуют и трехмерные игры, например Tux Racer. Большинство дистрибутивов включают в себя некоторые из этих игр, поэтому ваш выбор не ограничивается простыми пасьянсами и сапером. В состав KDE Desktop Environment вошли более 30 игр, включая Пасьянс, Нарды, клон игры Минер, Тетрис-подобную игру и Покер.

Таким образом, если вы любите поиграть, то без труда найдете достаточно много игр под Linux, и возможно даже, у вас появится больше причин избавиться от второй игровой платформы, которую вы, может быть, сохранили специально для игр. В этой главе рассказывается о нескольких играх, разрабатывавшихся специально для Linux, включая инструкции по установке, игровым моментам и, если это применимо, по запуску игрового сервера.

Quake III

Серия Quake долгое время была фаворитом среди фанатов игр-стрелялок за ее простой, но все же захватывающий сюжет и за ее графику. Хотя Quake и Quake 2 в первую очередь предназначались для одного игрока, обе они стали очень популярными в жанре матчевых игр, проводимых в Сети. С появлением Quake III компания Id Software взяла за основу мир Quake и создала игру, определенно ориентированную на нескольких игроков. В Quake III остался режим единственного игрока, но он представляет собой серию матчевых игр, где против игрока играют один или более компьютерных противников. По мере развития событий в игре противники становятся все более сильными, а в заключительном раунде вы остаетесь один на один с невероятно точным противником. В любом случае режим единственного игрока – это отличная подготовка для игры с несколькими игроками в Сети.

Вся серия игр Quake для Linux доступна для скачивания с сайта *ftp.idsoftware.com*. Когда Quake III впервые вышла в свет, в ней отсутствовали двоичные файлы для Linux, но позднее они появились в виде специального выпуска игры. Если у вас нет специального выпуска, вы можете скачать программу установки

для Linux и использовать существующий компакт-диск с игрой в версии для Windows.

Установка

Чтобы установить Quake III под Linux, загрузите последнюю версию программы установки из каталога *ftp.idsoftware.com/idstuff/quake3/linux*. После того как файл будет загружен, сделайте его исполняемым с помощью команды `chmod +x filename` и запустите программу установки из консоли, только не забудьте перед этим зарегистрироваться как *root*. Примите все лицензионные соглашения, и затем перед вами появится основное окно программы установки (рис. 7.1). По умолчанию игра устанавливается в каталог */usr/local/games/quake3*. Инсталлятор, имеющийся на компакт-диске с игрой, сам копирует файлы с расширением *.pk3* на жесткий диск, но инсталлятор, загруженный из Интернета, этого не делает. Таким образом, если вы пользуетесь версией инсталлятора, загруженной из Интернета, смонтируйте компакт-диск с версией игры для Linux или Windows и скопируйте файл *pak0.pk3* из каталога *Quake3/baseq3* на компакт-диске в каталог */usr/local/games/quqke3/baseq3*. Кроме того, если у вас имеется компакт-диск Team Arena, можно смонтировать этот диск и скопировать *pak0.pk3* из каталога *Setup/missionpack* на CD в каталог */usr/local/games/quake3/missionpack*.

После того как установка Quake III будет завершена, щелкните на соответствующем ярлыке в меню KDE или GNOME либо введите команду `quake3` в консоли. Игра Quake III основана на работе с графической библиотекой OpenGL, таким образом, вам потребуется обеспечить аппаратную поддержку трехмерной графики с OpenGL. В отличие от версии для Windows, версия игры для Linux не требует наличия компакт-диска в приводе CD-ROM на момент начала игры. При первом запуске в домашнем каталоге пользователя будет создан каталог *.q3a*, где будут храниться настройки и сохраненные игры. При желании файл с настройками можно редактировать напрямую, но настройки можно менять, находясь в самой игре, через меню Setup главного экрана.

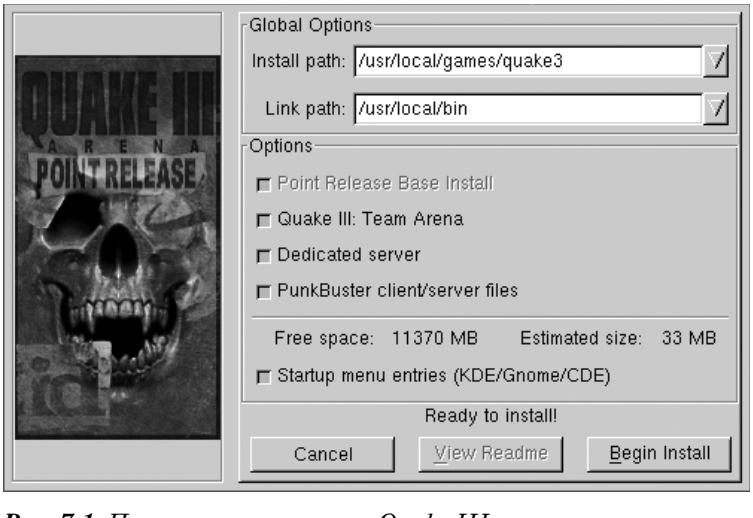


Рис. 7.1. *Программа установки Quake III*

Один игрок

В режиме одного игрока можно либо начать с первой арены и перемещаться по игре карта за картой, либо сразу перейти к нужному месту и щелкнуть на кнопке сражения в экране одного игрока. Сражение – это короткая миссия, которая не считается продвижением вперед в стандартной игре с одним игроком. Здесь можно выбрать типы и количество ботов противника, а также степень сложности. Режим одиночного сражения может служить тренировкой для оттачивания боевых навыков на определенной карте, это особенно полезно перед тем, как приступить к игре с несколькими игроками.

Правила ведения битвы достаточно просты – нужно убивать всех, кто попадается на глаза. С началом битвы персонаж появляется в одной из точек входа на карте. Суть битвы заключается в том, чтобы собрать как можно больше фрагментов или убить как можно больше противников до истечения заданного времени. По всей карте разбросаны оружие, боеприпасы, амуниция, аптечки и противники. В самом начале битвы игрок располагает простым автоматом и боевой рукавицей, поэтому желательно отыскать более мощное оружие, особенно важно успеть сделать это до встречи с противником (рис. 7.2). В самом начале персонаж обладает 100-процентным здоровьем, которое ослабевает с получением повреждений. Когда здоровье ослабевает до 0 или когда персонаж падает в одну из пропастей, он погибает и «возрождается» в одной из точек входа на карте. Все погибающие противники возрождаются аналогичным образом. Ограничений на количество «перерождений» нет, но следует помнить, что в этот момент теряются все оружие и амуниция, обретенные ранее, поэтому следует стараться погибать как можно реже.

Рис. 7.2. *Игра Quake III*

Серия игр Quake предоставляет стандартный набор видов оружия, из которого можно выбирать, и Quake III продолжает эту традицию, поддерживая бо̀льшую часть видов оружия, имевшихся в предыдущих версиях, и сохраняя их основные функции. Имеет смысл ознакомиться со всеми видами оружия поближе, потому что каждый из них обладает наибольшей эффективностью в определенных типах боя.

Gauntlet (боевая рукавица)

Стандартное оружие в игре, которое всегда имеется в наличии. Этот электризованный кулак не наносит противнику существенных повреждений и действует только на очень коротких дистанциях, но при этом не требует пополнения боеприпасов. Если противника удается уничтожить с помощью рукавицы, диктор объявит о нанесенном унижении, чтобы предупредить всех противников о факте победы с помощью такого слабого оружия.

Machine gun (автомат)

Оружие по умолчанию, имеющееся у игрока в самом начале. Автомат причиняет небольшие повреждения, но снабжается достаточно большим количеством боеприпасов и может быть эффективным оружием при высокой точности стрельбы.

Shotgun (дробовик)

Как и большинство дробовиков во многих подобных играх, этот дробовик вызывает серьезные повреждения при стрельбе с коротких дистанций. Чем дальше находится противник, тем меньшие повреждения наносятся ему при попадании.

Grenade launcher (гранатомет)

Очень эффективное оружие обороны. Гранатомет стреляет гранатами, которые взрываются или в момент контакта с противником, или после короткой задержки. Гранатомет удобно использовать, убегая от преследующего противника.

Rocket launcher (ракетная установка)

Ракетная установка – это мощное оружие, которое не только эффективно на средних и дальних дистанциях, но и способно уничтожать противников с первого попадания (и самого игрока, если он окажется слишком близко от места взрыва). Таким образом можно стрелять в стены и в пол около противника и при этом уничтожать его.

Lightning gun (электронная пушка)

Электронная пушка выстреливает в противника пучком молний. Замечательно действует на коротких дистанциях и способна удержать агрессивного противника на расстоянии.

Railgun (электромагнитная пушка)

Популярное оружие для снайперов. Электромагнитная пушка выстреливает пулей из обедненного урана. Один из недостатков этой пушки – невысокая скорострельность, на каждую перезарядку требуется примерно секунда времени. Об этом необходимо помнить, если электронная пушка используется в бою с быстро изменяющейся обстановкой.

Plasma gun (плазменная пушка)

Этот вид оружия выстреливает сгустки плазмы с очень коротким интервалом. Это оружие способно наносить повреждения даже при отсутствии прямого попадания, благодаря чему можно убить противника, даже не находясь в пределах прямой видимости.

BFG

BFG в Quake III несколько отличается от аналогичного оружия в Quake II. В Quake III BFG больше напоминает плазменную пушку увеличенной мощности. Выстреливает сгустками зеленой плазмы, которые вызывают очень серьезные повреждения, – это самое мощное оружие в игре.

Помимо оружия на картах можно найти дополнительные элементы, расширяющие возможности игрока. Можно найти аптечки и пакеты с боеприпасами, разбросанные по всей карте. На некоторых картах можно найти четырехкратный усилитель (Quad Damage). Когда игрок подбирает этот элемент, диктор объявляет: «Quad Damage», игрока охватывает голубое сияние, что позволяет увидеть его с большого расстояния, а эффективность всего имеющегося у него оружия на некоторое время увеличивается в четыре раза. Однако в случае гибели игрока элемент Quad Damage не исчезает вместе с ним, а остается в точке гибели, поэтому у остальных игроков появляется стимул уничтожить вас, чтобы завладеть этим усилителем.

Несколько игроков

Игра Quake III в первую очередь ориентирована на игру нескольких игроков. Режим с несколькими игроками мало чем отличается от режима одного игрока, за исключением того, что на этот раз придется сражаться не с компьютерными ботами, а с другими игроками. После выбора режима игры с несколькими игроками в главном меню откроется броузер сервера Quake III. Здесь можно просмотреть список доступных игровых серверов, к которым можно подключиться, а также карту, на которой в настоящее время идет игра, и количество участников. Можно задать конкретный сервер, введя его IP-адрес.

При желании можно создать свой собственный сервер, щелкнув на кнопке Create (Создать) и выбрав карту и количество ботов, после чего можно запустить сервер, щелкнув на кнопке Fight (Сражение). Если параметр Dedicated (Автономный) будет изменен перед запуском сервера, то он запустится в фоновом режиме, в противном случае игра начнется немедленно, и вы будете сразу же подключены к серверу. Если возникнет желание создать выделенный сервер для игры Quake III, прочитайте документ «Quake III HOWTO», который можно найти в Интернете, а для начала посетите страничку *http://www.planetquake.com/quake3/q3aguide/server-setup_a.shtml*.

Существует несколько типов игры с несколькими игроками, причем некоторые из них требуют установки в системе дополнительных модулей и карт. Основные типы игры – baseq3, стандартное сражение для нескольких игроков, и CTF – стандартный тип игры для двух команд, каждая из которых стремится захватить флаг другой команды. После того как вы найдете интересующую вас игру, к которой желаете присоединиться, щелкните на имени сервера, а затем на кнопке Fight (Сражение) внизу экрана.

Модули

Аналогично предыдущим версиям Quake, для Quake III сообществом любителей было создано большое количество дополнительных модулей. Некоторые из них просто добавляют новые виды оружия или типы игры, другие могут добавлять серьезные изменения в картах, оружии и даже изменять правила игры. Отыскать дополнительные модули, карты, интерьеры оформления можно на сайте *www.planetquake.com*.

Среди большого числа модулей и файлов можно найти очень популярный модуль Rocket Arena 3 (RA3) на страничке *http://www.planetquake.com/servers/arena/*. Модуль Rocket Arena 3 – это продолжение модуля Rocket Arena, имевшегося в Quake и Quake II. Он изменяет некоторые правила, что в результате дает неповторимый стиль игры. Прежде всего, по умолчанию в Rocket Arena 3 в самом начале игры персонаж располагает полным комплектом вооружения и боеприпасов. Это означает, что больше не нужно кружить по карте, пытаясь отыскать более мощное оружие. Далее, игрок не может получить повреждения от своего собственного оружия. Это означает, что можно выполнять некоторые маневры, недоступные ранее, такие как ракетный прыжок (запустить ракету себе под ноги и совершить прыжок на взрывной волне). Эти два изменения в правилах плюс полностью обновленный набор карт в сумме дают совершенно новую игру. На серверах RA3 можно выбрать игру против другого игрока в режиме «один на один» или в составе команды. В отличие от стандартной игры, здесь у игрока имеется всего одна жизнь, поэтому в случае гибели игрок должен дождаться окончания текущей игры, прежде чем он сможет присоединиться к ней.

Установка RA3 проходит всего в несколько шагов. Сначала нужно посетить сайт RA3 и отыскать программу установки для Linux/Mac, а затем загрузить *.zip*-архив размером 135 Мбайт. Далее распакуйте файл в каталог */usr/local/games/quake3/*, где в результате появится каталог *arena*. Чтобы приступить к игре в RA3, запустите Quake III из командной строки командой `quake3 +set fs_game arena` или выберите пункт arena в меню модулей. При желании запустить свой собственный сервер RA3 найдите сценарий *ra3server* в каталоге *arena*. Этот сценарий запускает автономный сервер RA3. Дополнительную информацию о настройке сервера RA3 вы найдете в каталоге *arena*, в файле *readsrv.txt*.

Return to Castle Wolfenstein

Для многих любителей компьютерных игр Wolfenstein 3D стала первой трехмерной игрой-стрелялкой. В игре вы американский солдат в нацистской тюрьме. Ваша цель состоит в том, чтобы выйти из тюрьмы и перестрелять всех охранников, которые встретятся на пути. Компания Id Software выпустила обновленную версию классического шутера под названием Return to Castle Wolfenstein (RTCW), где присутствуют практически те же элементы, но более обширное игровое поле с обновленной графикой и звуками.

К сожалению, клиент для Linux отсутствует на компакт-диске Return to Castle Wolfenstein, поэтому придется загрузить последнюю версию программы установки с сайта *ftp://ftp.idsoftware.com/idstuff/wolf/linux*. Здесь же, в этом же каталоге, находятся обновления, поэтому следует внимательно выбирать инсталлятор (это будет больший файл, не имеющий слова «update» в своем имени).

Установка

Для установки RTCW необходимо, обладая правами суперпользователя, сделать файл инсталлятора исполняемым с помощью команды `chmod +x filename` и запустить его. В результате на экране появится окно, где можно будет выполнить некоторые настройки, в том числе и выбрать каталог установки (рис. 7.3). По умолчанию игра устанавливается в каталог */usr/local/games/wolfenstein*.

В отличие от инсталляторов некоторых других игр под Linux, программа установки Wolfenstein не копирует сжатые файлы с компакт-диска на жесткий диск. В файле *README*, входящем в состав комплекта файлов программы установки, напрямую говорится о необходимости переписать шесть файлов из установленной версии игры для Windows в каталог */usr/local/games/wolfenstein/main*. Это файлы *mp_pak0.pk3*, *mp_pak1.pk3*, *mp_pak2.pk3*, *pak0.pk3*, *sp_pak1.pk3* и *sp_pak2.pk3*. Если игра для Windows не была установлена, можно попробовать запустить установку Windows-версии под управлением Wine, а затем скопировать требуемые файлы. Если в процессе установки возникнут какие-нибудь проблемы, обращайтесь к официальному списку вопросов и ответов для пользователей Linux, который можно найти по адресу *zerowing.idsoftware.com/linux*.

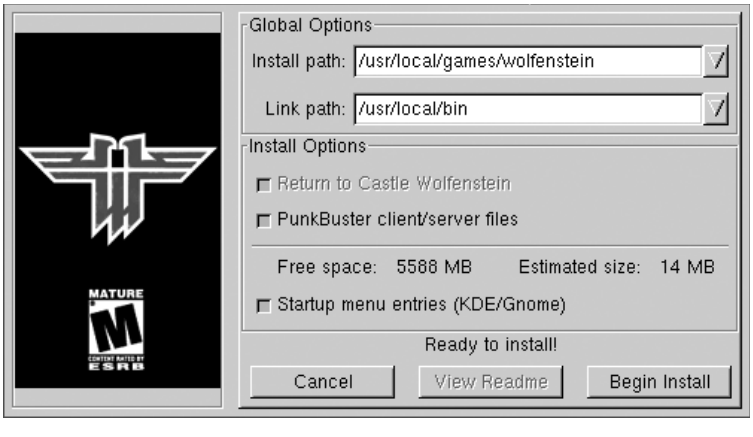


Рис. 7.3. *Программа установки RTCW*

Один игрок

После запуска игры можно щелкнуть на кнопке Options (Параметры) и ознакомиться с привязками клавиш управления по умолчанию (и изменить их), а также выполнить другие настройки по своему усмотрению. Кроме того, в разделе настроек можно запустить отдельные установленные модули. Чтобы запустить игру, необходимо щелкнуть по кнопке Play (Играть). В игре RTCW режимом по умолчанию является режим одного игрока, игра в котором начинается с ситуации, аналогичной первой версии Wolfenstein 3D: персонаж нападает на охрану и освобождается из тюремной камеры (рис. 7.4). Цель игры состоит в том, чтобы, преодолев все препятствия, выйти из тюрьмы.

По мере развития игровой ситуации можно будет двигаться бесшумно, замедлив скорость (надо нажать клавишу Caps Lock), осматриваться вокруг (Q и E, поворот

Рис. 7.4. *Начало игры RTCW*

влево и вправо, соответственно) или быстро, хотя и шумно, перемещаться через открытые пространства (удерживая клавишу Shift во время движения), открывать двери ногой (клавиша X) и перепрыгивать через препятствия (пробел). По ходу движения будут попадаться некоторые элементы, такие как оружие или припасы; чтобы подобрать элемент, достаточно лишь пройти по нему. Некоторые объекты в игре, как, например, устройства сигнализации, могут быть активированы и деактивированы клавишей Enter. Кроме того, устройства сигнализации могут быть деактивированы выстрелами из оружия.

В игре имеется масса оружия, которое поможет в борьбе с врагом. С самого начала игрок имеет обычный нож, но тут же может подобрать пистолет у убитого охранника. По мере обретения навыков и опыта игрок получает все более мощное оружие: автоматы, винтовки, гранаты и ракеты. Некоторые типы оружия используют один и тот же тип боеприпасов, поэтому необходимо постоянно следить за их расходованием и пополнением. Кроме того, некоторые из автоматических видов оружия быстро перегреваются, поэтому огонь из них желательно вести короткими очередями. На определенных уровнях имеются станковые пулеметы. Чтобы воспользоваться этим оружием, необходимо приблизиться к нему так, чтобы на экране появился значок с изображением руки, и затем нажать клавишу F или Enter, чтобы установить или снять пулемет. Когда пулемет будет установлен, вы сможете наблюдать окружающую обстановку через перекрестье прицела и стрелять в ваших врагов.

Несколько игроков

В RTCW имеется режим для игры нескольких игроков, который существенно отличается от игры с одним игроком. В режиме с несколькими игроками имеются две команды – Axis (противники) и Allies (союзники). По умолчанию в режиме с несколькими игроками обе команды имеют одну или более целей, которых они должны достигнуть за определенное время. Первая группа, достигшая цели, выигрывает раунд. В режиме состязания на время игра несколько видоизменяется. После каждого раунда игры группы меняются местами и должны постараться превзойти время друг друга. В режиме обхода контрольных пунктов игра напоминает захват флага. На игровом поле разбросано множество флажков. Побеждает та команда, которая первой соберет все флажки. Если в контрольное время не уложилась ни одна из команд, победившей считается та, которая наберет большее количество флажков.

В режиме с несколькими игроками, в отличие от режима одного игрока, есть возможность выбирать тип персонажа из четырех имеющихся, каждый из которых обладает различными навыками и боевым опытом. Успех команды обеспечивается взаимодействием персонажей всех типов.

Soldier (солдат)

Стандартная боевая единица. Солдат может пользоваться любым ручным оружием, например винтовкой. Этот тип будет лучшим выбором, если у вас еще нет опыта игры и вы не знаете, с чего начать.

Engineer (инженер)

Главная задача инженера – разрушение. Инженер – единственный игрок, который способен пользоваться взрывчаткой для разрушения объектов. Кроме того, инженер способен ремонтировать стационарные пушки и производить разминирование.

Medic (медик)

Медик – один из наиболее важных членов команды. Он может заживлять раны и даже реанимировать убитых членов команды. В экипировку медиков входят пакеты первой помощи, которые они могут передавать своим товарищам. Если в команде имеется медик, каждый член команды дополнительно получит 10% процентов здоровья. Кроме того, здоровье самого медика в случае получения им повреждений восстанавливается самостоятельно, хотя и на протяжении достаточно длительного периода времени.

Lieutenant (командир)

Главная цель командира заключается в выработке тактических решений. Командир имеет возможность поджечь дымовую шашку, чтобы сообщить об авианалете. У командира имеется бинокль, и он может вызвать огонь артиллерии на определенный квадрат. Командиры могут передавать пакеты с боеприпасами своим товарищам тем же способом, каким медики передают аптечки.

В отличие от некоторых других многопользовательских тактических шутеров, в RTCW игроки обладают несколькими жизнями и могут возрождаться, но с некоторыми особенностями. Когда игрок погибает, он помещается в состояние «неопределенности» и вынужден ждать завершения периода восстановления. В течение этого времени он может изменить свою роль и оружие и даже перейти

в другую группу. Вместо того чтобы находиться в состоянии неопределенности, можно дождаться прихода медика, который выполнит реанимацию. В этом случае возврат в игру происходит немедленно, как только санитар закончит восстановительные процедуры.

Играть с несколькими игроками можно или в локальной сети, не используя игровой сервер, или с помощью игрового сервера в Интернете. При выборе игры с несколькими игроками вам будет предоставлена возможность броузера серверов отыскать нужный сервер или напрямую указать IP-адрес сервера. Кроме того, можно запустить свой собственный сервер. В состав файлов версии игры для Linux входит файл с именем *QUICKSTART*, который выполняет запуск отдельного сервера. Исполняемый файл сервера называется `wolfded`. Настраиваемые параметры игры могут передаваться ему через аргументы командной строки или с помощью файла настроек. Чтобы запустить сервер с базовыми настройками, необходимо выполнить команду:

```
$ wolfded +set com_hunkmegs 64 +set sv_maxrate 9000 +set com_zonemegs 32 \
  +set dedicated 2 +set sv_hostname "my server" +set g_motd "my motd" \
  +map mp_villiage
```

В состав установки входит сценарий смены карт: *main/rotate.cfg*. Можно в командной строке передать этот сценарий серверу, чтобы получить игровой сервер, который циклически меняет игровые карты:

```
$ wolfded +set com_hunkmegs 64 +set sv_maxrate 9000 +set com_zonemegs 32 \
  +set dedicated 2 +set sv_hostname "my server" +set g_motd "my motd" \
  +exec rotate.cfg +vstr m_rotate1
```

В отличие от стандартных режимов с одним или несколькими игроками, для запуска автономного сервера не требуется иметь ключ к компакт-диску.

Чтобы обновить версию RTCW, необходимо загрузить последнюю версию программы установки (или, чтобы сэкономить на трафике, файл с тем же номером версии и со словом *–update* в имени) с сайта *frtp.idsoftware.com* и выполнить его, обладая правами суперпользователя. На экране появится точно такое же окно, что и при первоначальной установке, и после щелчка на кнопке Install (Установить) более новые версии файлов будут записаны поверх существующих.

Unreal Tournament 2004

Разработчики некоторых игр-шутеров пытаются преодолеть тот стереотип, что игры этого жанра не требуют больших умственных затрат от игрока, а успех в них в большей степени зависит от быстроты реакции и скорости движений. Для этого в режим одного игрока они добавляют основную сюжетную линию. Сюжетная линия обычно заключается в том, что игрок помещается во враждебную среду, и это служит основанием для уничтожения всего, что движется. Некоторые любители игр могли бы поиграть в режиме одного игрока время от времени, но в большинстве своем они предпочитают тратить время на игру с другими людьми.

Игра Unreal Tournament (UT) не дает такого обоснования и целиком сосредотачивается на игре в стиле арены. Благодаря этому UT обрела массу поклонников среди любителей компьютерных игр в сети, поскольку включает множество раз-

нообразных карт арены и стилей сетевой игры, начиная от обычных сражений и сбора флажков до игр с бомбами и других, которые начинают размывать грань между обычными шутерами и играми спортивного жанра.

Установка

Как и выпущенные ранее Unreal Tournament и Unreal Tournament 2003, Unreal Tournament 2004 (или UT2K4) обладает поддержкой Linux как в клиентской, так и в серверной ее части. Однако, в отличие от многих игр других компаний, имеющих реализацию для Linux, UT2K4 поставляется в виде комплектов двоичных файлов с инсталлятором для Linux на одном диске, точно так же, как и пакет для Windows. Это означает, что нет необходимости искать в Интернете подходящее зеркало и загружать с него большой *.sh*-файл или ждать недели и месяцы, когда выйдет версия для Linux, – вы можете сразу же приступить к игре.

Каталог установки UT2K4 выбирается в зависимости от того, кто запустил установку. Если это обычный пользователь, UT2K4 создает каталог *ut2004* в домашнем каталоге пользователя. Игра отлично работает в любом случае независимо от того, кто ее установит, однако если есть возможность получить привилегии суперпользователя, лучше установить ее сразу для всех пользователей системы; это пригодится на тот случай, если позднее будет принято решение об отключении пользователя. Если сценарий *linux-installer.sh* будет запущен с привилегиями суперпользователя, он установит игру в каталог */usr/local/games/ut2004/* по умолчанию и сделает его доступным для всех пользователей в системе.

Чтобы установить UT2K4, смонтируйте первый компакт-диск (с меткой «Install Disc») и запустите файл с именем `linux-installer.sh`. В большинстве файловых менеджеров это можно сделать простым щелчком на имени файла. Если у вас этого не происходит, откройте терминал и введите команду:

```
# /mnt/cdrom/linux-installer.sh &
```

После того как будут приняты условия лицензионного соглашения, вы увидите первый экран программы установки, как показано на рис. 7.5.

В этом экране можно изменить многие параметры установки, включая каталог установки, язык и добавление элементов меню для KDE и GNOME. Для установки потребуется примерно 5 Гбайт дискового пространства, поэтому необходимо убедиться в наличии такого объема свободного пространства перед тем, как начать установку. В случае нехватки свободного места кнопка Begin Install (Начать установку) окажется заблокированной. После щелчка на кнопке Begin Install (Начать установку) программа установки попросит ввести ключ компакт-диска и затем начнет копировать файлы с CD-ROM на жесткий диск. Если была приобретена версия игры UT2K4 на компакт-дисках, в процессе установки будет предложено вставить другой диск. Если была приобретена версия игры на DVD, установка пройдет без необходимости менять диски.

По окончании установки в последнем окне можно будет запустить игру, щелкнув на кнопке Start (Пуск). Кроме того, запустить игру можно будет, выбрав соответствующий пункт в главном меню KDE или GNOME либо просто введя команду `ut2004` в командной строке. В отличие от версии для Windows, чтобы иметь возможность играть, не обязательно монтировать установочный компакт-диск.

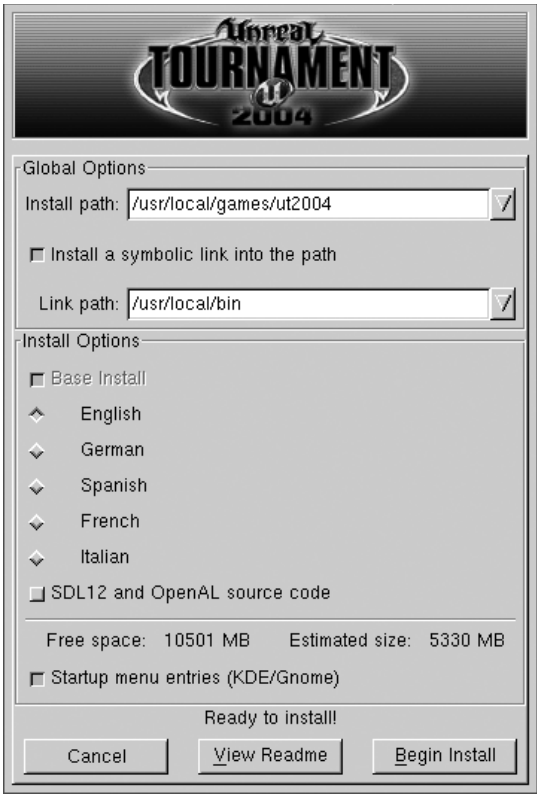


Рис. 7.5. *Первый экран программы установки UT2K4*

Игра

Самое первое, что было бы желательно сделать сразу же после запуска игры, – это ознакомиться со всеми ее настройками и привязками клавиш. Все необходимые настройки могут быть выполнены в меню Settings (Настройки) из главного экрана игры. Здесь можно настроить все параметры игры, начиная от разрешения экрана и заканчивая специальными визуальными эффектами. Параметры настройки и состояние игры сохраняются в каталоге *~/.ut2004*. При желании файлы настроек (имеющие расширение *.ini*) в каталоге *~/.ut2004/System/* можно редактировать вручную.

UT2K4 имеет несколько вариантов игры. Первый – игра в режиме одного игрока, в этом режиме человек выступает в качестве капитана команды бойцов, которых он набирает сам. Затем группа начинает повышать свой рейтинг, участвуя в командных играх различного типа:

Capture the Flag (захват флага)

Традиционный вариант захвата флага, имеющийся в других шутерах. Ваша группа должна проникнуть на базу противника, захватить его флаг и вер-

нуться к своему флагу, одновременно препятствуя противнику сделать то же самое. Очки начисляются за каждый захваченный флаг.

Bombing Run (игра с бомбами)

Данная игра немного напоминает баскетбол. В середине карты помещается мяч, на стороне каждой команды имеется мишень. Очки начисляются, когда игрок подбирает мяч и поражает им мишень противника. Если игрок с мячом будет убит, мяч останется на месте гибели и его можно будет подобрать.

Double Domination (двойное превосходство)

Вариант игры по захвату флага. В этой игре на карте имеются два «пункта превосходства», которые изначально являются ничьими. Цель игры состоит в том, чтобы пробежать через пункт превосходства, сделав его своим, и удержать оба пункта за собой. За каждые 10 секунд, в течение которых пункты остаются вашими, начисляется одно очко.

Assault (нападение)

Действие игры происходит в несколько раундов. В каждом раунде одна из команд принимает на себя роль нападающей стороны, а другая – обороняющейся. Группе нападающей стороны необходимо достигнуть нескольких целей за ограниченное время (например, определенного местоположения на карте), и группа обороняющихся должна остановить их. По окончании раунда группы меняются ролями.

В режиме игры одного игрока удобно знакомиться с различными картами и типами игры, поскольку в этом случае сложность игры нарастает медленно. В каждой точке игры можно выполнить сохранение ее состояния, поэтому в следующий раз можно начать с того места, где игра была остановлена. Если вы хотите потренироваться и получить боевые навыки на определенной карте или с определенным типом игры, попробуйте выбрать режим Instant Action (Одиночная битва). В этом случае вы сможете выбрать тип игры и карту, а также количество ботов и уровень сложности. В одиночных битвах можно оттачивать свои навыки на определенной карте или просто попрактиковаться, не рискуя повлиять на свой общий счет побед и проигрышей.

Постарайтесь ознакомиться с каждым типом оружия и их возможными режимами стрельбы, так как некоторые виды оружия позволяют стрелять разными боеприпасами, например ударная винтовка в альтернативном режиме может посылать сгустки плазмы, по которым можно вести огонь в штатном режиме, чтобы вызвать детонацию плазмы и взрыв большой разрушительной силы (рис. 7.6). Наибольшего успеха добиваются те игроки, которые мастерски владеют различными типами оружия и могут в случае необходимости быстро переключаться между ними.

Если вы готовы сыграть против других игроков в Интернете или в локальной сети, выберите вариант Join Game (Присоединиться к игре). После этого вам будет предоставлена возможность выбрать область поиска – в локальной сети или в Интернете, а затем UT2K4 отыщет и выведет список всех обнаруженных игр. Чтобы присоединиться к игре, достаточно просто щелкнуть на кнопке Join (Присоединиться), но помните, чем меньше время отклика сервера, тем более динамичной будет игра.

***Рис. 7.6.** Ударная винтовка в UT24K*

Игровой сервер

Имеется возможность запустить свой сервер игры UT2K4, выбрав пункт Host Game (Запустить игровой сервер). Здесь можно будет настроить собственный игровой сценарий, во многом напоминающий Instant Action (Одиночная битва), что даст возможность выбрать карты, определить число игроков, число ботов и другие параметры, а затем запустить собственный сервер, к которому смогут присоединиться другие участники, находящиеся в локальной сети или в Интернете. Как только настройка сервера будет завершена, можно щелкнуть на кнопке Listen (Принимать подключения) или Dedicated (Автономный), чтобы запустить сервер. Щелчок на кнопке Listen (Принимать подключения) запускает сервер и немедленно подключает вас к нему. Щелчок на кнопке Dedicated (Автономный) запускает сервер в фоновом режиме, но не подключает вас к нему. Это удобно, когда необходимо запустить игровой сервер, но сами вы не планируете принимать участие в игре. Кроме того, игровой сервер можно запустить прямо из командной строки. Для этого нужно перейти в каталог *ut2004/System* и запустить сценарий `ucc` с аргументами `server` и именем выбранной карты:

```
# cd /usr/local/games/ut2004/System
# ./ucc server DOM-SunTemple
```

Одно из преимуществ, которое дает подобный запуск сервера, состоит в том, что серверы могут работать на машинах, не имеющих ускорителя трехмерной графики и поддержки графики вообще. Все параметры, которые можно изменить с помощью графического интерфейса, могут быть также изменены и из командной строки, а в Интернете можно найти немало руководств, которые описывают назначение различных параметров командной строки. Если вам уже приходи-

лось заниматься настройкой сервера Unreal Tournament предыдущих версий, вы обнаружите, что большинство настраиваемых параметров остались теми же самыми и в Unreal Tournament 2004.

Обновления

Если вы предполагаете вести игру в Интернете, вам придется постоянно следить за выходом обновлений. Иногда обновления не просто устраняют ошибки – они предотвращают возможность обмана. Объявления о выходе новых исправлений размещаются на официальном сайте Unreal Tournament *www.unrealtournament.com*, но о выходе исправлений для Linux можно узнать еще и на сайтах *www.icculus.org* и *www.linusgames.com*. Щелкните на пункте Join Game (Присоединиться к игре) и посмотрите в правом верхнем углу номер версии игры.

Обновления выкладываются в формате *.tar.bz2*, и потому, чтобы применить их, сначала требуется распаковать архив в отдельный каталог и затем переписать файлы обновления поверх существующих:

```
# tar -xjf ut2004-lnxpatchversion.tar.bz2
# cd UT2004-Patch
# /bin/cp -a * /usr/local/games/ut2004/
```

Эмуляторы

Современные игры с подробной графикой, сложной музыкой и возможностью игры в сети – все это, конечно, увлекательно, но иногда возникает желание вернуться в прошлое, во времена 8- или 16-разрядной графики, когда сам игровой сюжет был гораздо важнее, чем качественная графика. Для Linux существует множество эмуляторов, что дает вам возможность вспомнить дни, когда появилась игра Pacman. Эти эмуляторы работают с образами картриджей для игровых приставок и эмулируют среду, в которой образ сможет нормально функционировать, что позволяет использовать клавиатуру или даже джойстик и запускать игры непосредственно на компьютере.

MAME

Пожалуй, самым известным и популярным эмулятором игровых приставок считается MAME (Multiple Arcade Machine Emulator – эмулятор нескольких игровых приставок). Основная цель проекта MAME (*www.mame.net*) заключается в создании эмулятора различных игровых платформ, которые пользовались популярностью в прошлые годы. В этом смысле проект MAME намного более сложен, чем другие подобные проекты, потому что поддерживает большое количество различных платформ. В настоящее время MAME поддерживает тысячи различных названий игр, и этот список продолжает увеличиваться. Полный список названий можно найти на страничке *www.mame.net/gamelist.html*.

Изначально проект MAME поддерживал только платформу Windows, однако позднее появилась реализация для Linux, которая получила название Xmame. Она основана на исходных текстах MAME с изменениями, которые необходимо было внести, чтобы эмулятор мог работать под управлением Linux. Можно смело считать, что Xmame под Linux – это все равно что MAME под Windows, поэтому образы картриджей, работающие в MAME, будут также работать и в Xmame.

Xmame – это достаточно популярная программа, и потому она должна входить в состав большинства дистрибутивов. Если же в вашем дистрибутиве ее не окажется, последнюю версию в исходных текстах можно загрузить с сайта проекта *x.mame.net*. Xmame поддерживает целый ряд параметров работы с графикой, и в некоторых дистрибутивах эта программа может поставляться в виде отдельных пакетов, различающихся принципом работы с графикой:

X11

Стандартный режим, в котором вывод графики на экран производится средствами X Window System.

SVGALib

Xmame поддерживает возможность вывода графики в режиме текстовой консоли, что дает возможность работать с Xmame в системах, не имеющих поддержки X.

GL

Вывод графики на экран может осуществляться средствами X11 с помощью библиотек OpenGL, что позволяет использовать возможности аппаратного графического ускорителя.

SDL

Xmame может также использовать библиотеки SDL для вывода графики в окружении X11. Аналогично библиотекам OpenGL, библиотеки SDL позволяют использовать возможности аппаратного графического ускорителя.

Glide

Аналогично OpenGL, Xmame может использовать библиотеки Glide для поддержки возможностей аппаратного графического ускорителя как под управлением X, так и под управлением SVGALib для видеокарт 3DFX.

Для начала стоит попробовать использовать режим вывода графики средствами X11, потому что этот метод установлен по умолчанию и наверняка будет работать на большинстве систем. Сначала просто запустите команду `xmame` из командной строки без дополнительных аргументов. Программа отыщет файл */etc/xmame/xmamerc* с глобальными настройками по умолчанию, создаст локальный каталог с настройками *~/.xmame* и скопирует файл */etc/xmame/xmamerc* в каталог *~/.xmame*, что даст возможность выполнить настройки, специфичные для конкретного пользователя. В этом файле находятся все необходимые параметры настройки Xmame. Самое первое, что потребуется изменить в данном файле, – это путь к каталогу с образами картриджей (`rompath`). Этот параметр определяет пути поиска файлов-образов, поэтому, если ваши образы хранятся в локальном каталоге, добавьте путь к каталогу в этот параметр и сохраните файл. После этого можно попробовать поиграть в любую игру, образ которой у вас имеется. Для этого нужно запустить программу, передав ей имя файла-образа в качестве аргумента. Обычно файлы образов распространяются в виде *zip*-архивов, в которых находятся несколько файлов, необходимых эмулятору. Например, чтобы начать игру с образом *pacman.zip*, достаточно запустить команду

```
$ xmame pacman
```

из командной строки (рис. 7.7).

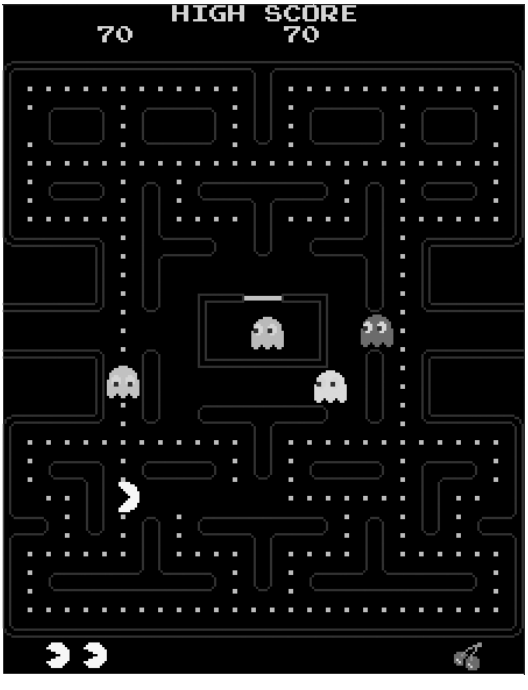


Рис. 7.7. *Игра Pacman в Xmame*

Теперь, когда игра запущена, как управлять ею? В Xmame используются те же самые привязки клавиш, что и в версии MAME для DOS. В табл. 7.1 приводятся некоторые из привязок, которые можно использовать.

Таблица 7.1. Привязки клавиш в Xmame

Клавиша	Действие
P	Пауза
F3	Сброс игры
F8	Уменьшить количество пропускаемых кадров
F9	Увеличить количество пропускаемых кадров (удобно, когда хочется поиграть в быструю игру на медленной машине)
Esc	Выход из эмулятора
Левая клавиша Shift+PageUp	Увеличить окно (для дисплеев с высоким разрешением)
Левая клавиша Shift+PageDown	Уменьшить окно
Левая клавиша Shift+Insert	Обычный размер окна
Левая клавиша Shift+Home	Полноэкранный режим DGA

Кроме этих, в играх имеются собственные привязки клавиш. Они могут отличаться в разных играх, но для большинства игр существует стандартный набор привязок (табл. 7.2).

Таблица 7.2. Стандартные привязки клавиш

Клавиша	Действие
Клавиши управления курсором	Перемещение влево, вправо, вверх и вниз
1	Выбор режима с одним игроком
2	Выбор режима с двумя игроками
5	Вставить монету
Ctrl	Клавиша 1
Alt	Клавиша 2

Поскольку некоторые игровые приставки имеют только джойстик, а другие до шести или более клавиш, привязки клавиш на клавиатуре могут несколько изменяться. В простых играх клавиши Ctrl и Alt воспринимаются как первая и вторая клавиши, но в случае с более сложными играми может потребоваться некоторое время, чтобы обнаружить имеющиеся привязки. По умолчанию игры запускаются в Xmame с оригинальным разрешением, поэтому для дисплея с высокой разрешающей способностью может потребоваться увеличить масштаб игры в два или три раза с помощью комбинации «левая клавиша Shift»+PageUp.

В некоторых играх Xmame позволяет использовать мышь. В качестве прекрасного примера игровой приставки, где используется мышь, можно привести Centipede, в состав которой входит большой трекбол.

Кроме того, Xmame обладает поддержкой джойстика, хотя по умолчанию она выключена. Чтобы включить поддержку джойстика, необходимо изменить параметр `joytype` в файле *xmamerc* или передать программе параметр командной строки `-joytype` *число*, где *число* определяет тип джойстика (табл. 7.3).

Таблица 7.3. Типы джойстиков

Число	Тип джойстика	Число	Тип джойстика
0	Нет джойстика	4	Новый драйвер джойстика i386 для Linux
1	Драйвер джойстика i386	5	Драйвер джойстика USB для NetBSD
2	Поддержка Fm Town Pad	6	Клавиатура PS2-Linux
3	Джойстик, поддерживаемый расширениями ввода X11	7	Драйвер джойстика SDL

Nestra

Видеоигры невероятно увлекательны, но есть такие игры, которые, похоже, существуют только для приставок определенного типа, таких как Nintendo Entertainment System (NES). Однако существует программное обеспечение – аналог MAME, – которое эмулирует аппаратуру NES и позволяет запускать игры для

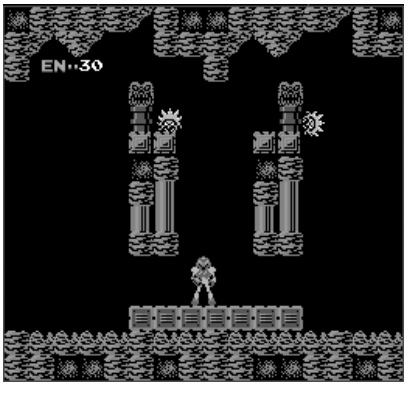


Рис. 7.8. *Игра Metroid под управлением Nestra*

этой приставки на компьютере. Это программное обеспечение для операционной системы Linux называется Nestra.

Скорее всего, Nestra также входит в состав вашего дистрибутива. Если это не так, данную программу можно загрузить в исходных текстах с сайта *nestra.linuxgames.com*. Работа с Nestra после ее установки не должна вызывать сложностей. Запускается она командой `nestra`, которой передается путь к файлу образа в виде аргумента. Например, чтобы запустить оригинальную игру Metroid (рис. 7.8), необходимо ввести команду:

```
$ nestra Metroid.nes
```

Привязка клавиш в Nestra остается неизменной для всех игр и соответствует стандарту контроллера NES (табл. 7.4).

Таблица 7.4. Привязки клавиш в Nestra

Клавиша	**Функция**
Клавиши управления курсором	Клавиши со стрелками
Пробел	Клавиша A
z, x	Клавиша B
Enter	Пуск (Start)
Tab	Выбор (Select)
Pause, Break	Сброс (Reset)
Esc	Выход из программы Nestra
1–9	Коррекция скорости эмуляции: 1 – нормальная скорость, 2 – двойная, и т. д.
-	Скорость в 2 раза ниже нормальной
0	Пауза

Некоторые игры, например Zelda, позволяют сохранять состояние игры в самом картридже. Поскольку в данном случае мы имеем дело с эмулятором ПЗУ, со-

храненная игра записывается в файл в тот каталог, где находится файл образа, или в каталог *~/.nestra*, если таковой существует.

SNES9x

Приставка NES дала нам действительно замечательные игры, а когда появилось следующее поколение приставки Nintendo – Super Nintendo Entertainment System, или SNES, – многие игры, ставшие классикой, такие как Super Mario Bros, Zelda или Metroid, перекочевали на новую платформу. Под управлением Snes9x будут работать все ваши любимые игры, выпущенные для SNES.

Snes9x – это эмулятор SNES, который имеет реализации для Windows, Linux, Mac OS X и других платформ. Реализация для Linux входит в состав большинства дистрибутивов, но при желании ее можно загрузить в исходных текстах с официальной страницы проекта Snes9x – *www.snes9x.com*.

После установки работа с программой Snes9x практически не отличается от Nestra – просто введите в командной строке команду `snes9x` и путь к файлу образа в качестве аргумента. Например, чтобы запустить игру Zelda 3, достаточно ввести команду:

```
$ snes9x zelda3.smc
```

В отличие от Nestra, Snes9x имеет гораздо больше возможных параметров, которые можно передать программе в командной строке. Например, параметр *-y* включает «телевизионный режим», который увеличивает изображение в два раза и вставляет дополнительный пиксел между строками развертки. Назначение телевизионного режима – получить большое изображение, напоминающее обычный телевизионный экран.

В Snes9x используются стандартные привязки клавиш, соответствующие клавишам на контроллере SNES (табл. 7.5).

Таблица 7.5. Привязки клавиш в Snes9x

Клавиша	Функция	Клавиша	Функция
Escape	Завершение работы эмулятора	s, m, e	Клавиша X
Pause, Scroll Lock	Пауза	x, ',', r	Клавиша Y
Стрелка вверх, u	Движение вверх	d, '.', t	Клавиша A
Стрелка вниз, j, n	Движение вниз	c, y	Клавиша B
Стрелка влево, h	Движение влево	Enter	Клавиша Пуск (Start)
Стрелка вправо, k	Движение вправо	Пробел	Клавиша выбора (Select)
a, v, q	Клавиша TL	Shift+F1–F9	Сохранить игру в одном из девяти слотов
z, b, w	Клавиша TR	F1–F9	Загрузить сохраненную ранее игру из заданного слота

Эмулятор Snes9x имеет поддержку джойстиков: по умолчанию это устройство *dev/js0*, но есть возможность определить другое устройство джойстика с помо-

щью параметра *-joydev1*. С помощью параметров *-joymap1* и *-joymap2* можно также управлять отображением восьми клавиш SNES (для первого и второго джойстиков соответственно) на клавиши компьютера. Например, значение по умолчанию – 0 1 2 3 4 5 6 7, что соответствует клавишам A B X Y TL TR Start Select.

Программа Snes9x имеет так много возможных параметров, что для упрощения их настройки были разработаны отдельные программы с графическим интерфейсом. Программа Snes9express предоставляет простой и удобный в работе интерфейс, облегчающий управление коллекциями образов и различными параметрами настройки. Последнюю версию программы можно загрузить с сайта *www.linuxgames.com/snes9express* или воспользоваться пакетом, входящим в состав дистрибутива. Программа Snes9express поддерживает возможность изменения внешнего оформления интерфейса, причем одна из тем даже напоминает внешний вид пульта управления приставки SNES (рис. 7.9).

Выберите пункт меню Console (Консоль)→Preferences (Настройки) и укажите в настройках путь к каталогу с файлами образов картриджей для SNES. Затем щелкните на кнопке Selector ROM, и на экране появится окно со списком всех имеющихся игр. Выберите желаемую игру и щелкните на кнопке Power (Пуск), чтобы запустить ее. После этого окно Snes9express исчезнет и появится окно с запущенной игрой. Как только игра будет закончена, окно со списком игр опять появится на экране.

Чтобы изменить параметры настройки Snes9x, находясь внутри Snes9express, выбирайте различные вкладки с параметрами настройки в основном окне. На

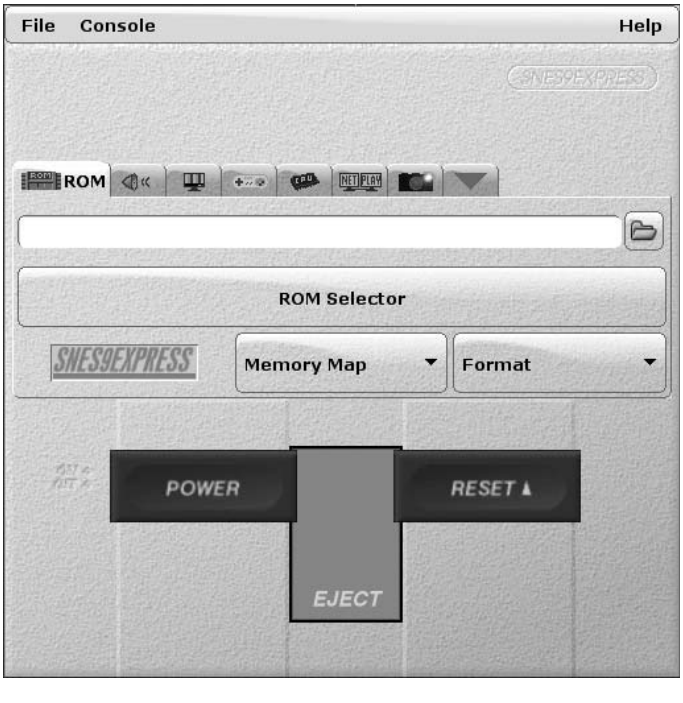


Рис. 7.9. *Snes9express*

вкладках вы найдете настройки звука, видео, контроллера и других параметров, а графический интерфейс облегчит их изменение.

Frozen Bubble

Некоторые игры, несмотря на свою простоту, обладают необычайной притягательностью – к ним хочется возвращаться снова и снова. Frozen Bubble – это игра-головоломка, напоминающая Puzzle Bubble или Bust-a-Move. Цель игры Frozen Bubble состоит в том, чтобы убрать все разноцветные пузырьки, расположенные в верхней части экрана (рис. 7.10). Игроку дается единственный цветной пузырек в нижней части, и он должен, как следует прицелившись, поразить пузырек того же цвета в верхней части экрана. Если удастся попасть в пузырек того же цвета, то этот и все другие пузырьки, связанные с ним, исчезнут. Если поразить пузырек не удастся, выпущенный вами пузырек станет одним из тех, которые надо поразить. Расстреляв все пузырьки, игрок переходит на другой уровень сложности. Пузырьки, расположенные сверху, постепенно опускаются вниз, поэтому, если не удастся достаточно быстро расстрелять все пузырьки, они достигнут нижнего края и игрок будет считаться проигравшим.

Игра Frozen Bubble известна достаточно широко и наверняка входит в состав вашего дистрибутива. Но ее можно загрузить в исходных текстах с официального сайта *www.frozen-bubble.org*. Запустить игру можно или с помощью меню рабочего стола, или командой `frozen-bubble`. Игра может быть запущена в режиме одного или двух игроков и даже позволяет создавать свои собственные уровни с помощью редактора уровней.

Рис. 7.10. *Игра Frozen Bubble*

В режиме с одним игроком игра ведется на время. Управление игрой простое и легкое. Клавиши со стрелками влево и вправо перемещают прицел влево и вправо, соответственно, а клавиша со стрелкой вверх запускает пузырек, находящийся внизу. Чтобы поразить цель в труднодоступном месте, можно воспользоваться тем обстоятельством, что пузырек отскакивает от стен. Иногда удается даже пройти уровень одним-единственным выстрелом.

В режиме двух игроков оба участника играют одновременно (рис. 7.11). Оба игрока управляют игрой с клавиатуры, поэтому первый игрок перемещает свой прицел влево и вправо клавишами X и V, соответственно, и производит выстрел клавишей C. Второй игрок использует клавиши со стрелками. Первый, кто поразит все цели, считается победителем.

Включенный в поставку редактор позволяет создавать свои варианты уровней, для которых можно задавать количество, цвет и местоположение пузырьков. Щелчок правой кнопкой мыши удаляет пузырек, а левой – выбирает цвет пузырька. Редактор позволяет изменить любой из имеющихся 100 уровней.

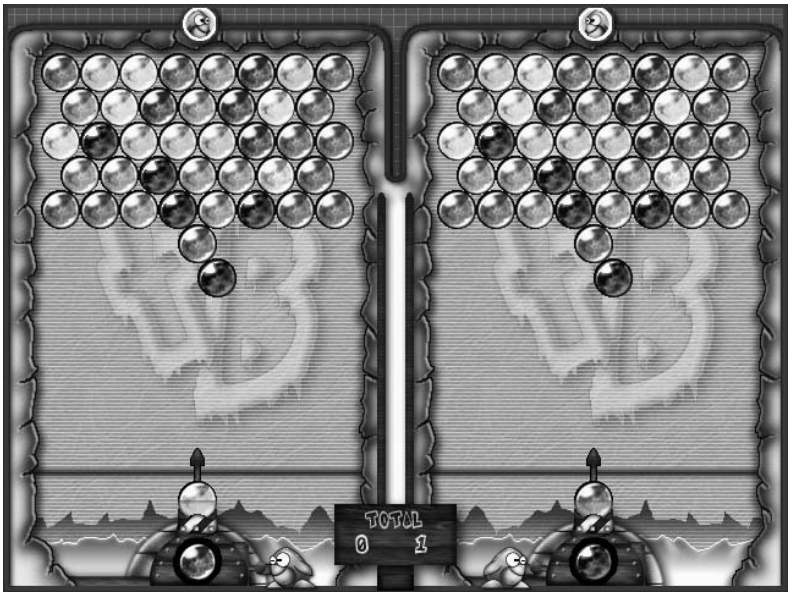


Рис. 7.11. *Игра Frozen-Bubble в режиме двух игроков*

Tux Racer

Что бы это была за глава, если бы в ней не упоминалась игра, главным персонажем которой является талисман операционной системы Linux – пингвин Тукс (Tux). Tux Racer – это трехмерная игра-гонка, но вместо автомобилей или других транспортных средств игрок управляет пингвином, скатывающимся на животе вниз по склону ледяной горы. Успех зависит от того, насколько быстро завершится гонка и сколько рыбин удастся съесть по пути.

Игра Tux Racer начиналась как проект с открытыми исходными текстами вплоть до версии 0.61. Добившись определенного успеха и претерпев значительные усовершенствования, Tux Racer 1.0 была выпущена студией Sunspire Studios в виде самостоятельного коммерческого продукта. Коммерческую версию Tux Racer 1.0 можно приобрести на официальном сайте *www.tuxracer.com*, но пока еще в свободном доступе имеется версия 0.61 с открытыми исходными текстами, которую можно загрузить с сайта *tuxracer.sourceforge.net*. Именно эта версия обычно включается в состав большинства дистрибутивов.

Запустить игру Tux Racer можно или с помощью меню, или введя `tuxracer` в командной строке. После запуска игры на экране появляется начальное меню, где можно выбрать один из вариантов игры – практика или соревнование. Соревнование – это серия гонок, в которой каждая гонка требует, чтобы были пройдены предыдущие гонки. Выбирая доступные гонки, можно будет увидеть максимальное время и число рыбин, которые необходимы, чтобы перейти к следующей гонке. Если любое из тех требований не будет соблюдено, гонку придется пройти еще раз.

Игра отличается чрезвычайно простым управлением, но для точного управления игрой необходим некоторый навык. Клавиши со стрелками влево и вправо управляют направлением движения Тукса. Клавиша со стрелкой вверх заставляет Тукса махать крыльями, что вызывает разные эффекты в зависимости от того, в каком месте гонки вы находитесь. Если пингвин движется медленно (особенно в самом начале), это увеличивает его скорость. Когда показания спидометра войдут в желтую зону, движения крыльями замедляют скорость. Кроме того, во время прыжков движения крыльями позволяют Туксу дольше оставаться в воздухе и корректировать направление полета. Клавиша со стрелкой вниз служит тормозом. Комбинируя нажатие клавиши со стрелкой вниз с одновременными нажатиями клавиш со стрелками влево или вправо, можно выполнять более крутые повороты.

Тукс может также совершать прыжки с помощью клавиши `E`. Удержание этой клавиши заряжает энергетический потенциал (Energometer) Тукса – чем он выше, тем более высокий прыжок совершит Тукс. Если Тукс застрянет где-нибудь, его можно вернуть на трассу с помощью клавиши `Backspace`, а чтобы прервать игру, можно нажать клавишу `Q`.

Файлы игры Tux Racer хранятся в каталоге *~/.tuxracer*. Настраиваемые параметры игры находятся в файле *~/.tuxracer/options*, здесь можно определить, должна ли игра запускаться в полноэкранном режиме. В этом же файле можно переопределить привязки клавиш клавиатуры и джойстика, которые используются для управления игрой.

В режиме практики игра позволяет выбирать различные уровни сложности, не требуя прохождения предыдущих уровней или выполнения каких-либо других требований. Это позволяет выбрать любимый уровень и практиковаться в игре, не беспокоясь о времени или количестве собранных рыбин. На самом популярном уровне, который носит название «Кто сказал, что пингвины не умеют летать» («Who said penguin can't fly»), пингвин спускается с горы с максимально возможной скоростью (рис. 7.12).

Рис. 7.12. *Кто сказал, что пингвины не умеют летать?*

В игре есть определенные стратегии, позволяющие проходить гонки за минимальное время. Не по всем поверхностям Тукс скользит с одинаковой скоростью. Самая высокая скорость достигается при спуске по гладкому льду, чуть медленнее – по шероховатому снегу и совсем медленно – по незаснеженному грунту. Незаснеженный грунт настолько замедляет движение, что его следует по возможности избегать. Кроме того, двигать крыльями следует только до тех пор, пока показания спидометра не вошли в желтую зону, – ниже этой зоны взмахи крыльями ускоряют движение, выше – замедляют. Существенный прирост в скорости дают перелеты по воздуху. Участки трассы с поворотами можно использовать как своеобразные трамплины. На спуске Тукс может набрать высокую скорость, но следует избегать препятствий, которые замедляют движение. Кроме того, не следует забывать, что взлетая слишком часто, можно пропустить ценную рыбу, которую необходимо собирать по мере спуска.

8

Офисные пакеты и приложения личного пользования

За прошедшие годы операционная система Linux проделала долгий путь. Когда люди начали использовать Linux не только ради экспериментов с системой, но и для того, чтобы сделать что-то полезное, в разряд наиболее используемых приложений попали различные виды серверов: серверы электронной почты или веб-серверы. Обычные настольные приложения и приложения личного пользования, такие как текстовые процессоры, электронные таблицы или инструменты для совместной деятельности, практически не были известны в Linux.

В настоящее время ситуация кардинально изменилась. Появилось большое количество офисных пакетов и других приложений личного пользования, и в этой главе будет рассказано о некоторых из них. Основное внимание будет уделено пакету OpenOffice, пожалуй, самому полному на сегодня пакету офисных приложений для Linux, а также мы остановимся на некоторых других приложениях.

OpenOffice

На сегодняшний день OpenOffice является наиболее полнофункциональным бесплатным пакетом офисных приложений для GNU/Linux, распространяемым с открытыми исходными текстами. По умолчанию он входит в состав большинства дистрибутивов, включая SUSE, Red Hat, Debian и другие.[1]

Это совсем не означает, что не следует уделять внимание другим свободно распространяемым офисным пакетам, таким как KOffice и AbiWord. Это было бы совершенно неправильно, но тем не менее OpenOffice обладает некоторым преимуществом благодаря относительной зрелости программного кода, изначально встроенной поддержке открытого формата XML (этот формат поддерживается

[1] Более того, на официальном ресурсе проекта *http://www.openoffice.org/dev_docs/source/sys_reqs.html* представлены официальные версии OpenOffice для операционных систем: Microsoft Windows, GNU/Linux, Sun Solaris, Mac OS X (under X11) и FreeBSD. Кроме того, поскольку OpenOffice предоставляется в исходных кодах, он может быть импортирован заинтересованными сообществами и в другие, более экзотические операционные системы. – *Примеч. науч. ред.*

и пакетом KOffice), а также способности пакета работать под управлением операционной системы Windows и его совместимости с популярными патентованными форматами документов.

Термины «OpenOffice» и «OpenOffice.org»

Чтобы не запутаться в терминологии, которая будет использоваться далее в этой главе, необходимо прийти к некоторым соглашениям. Термин «OpenOffice», или его сокращенная форма «OOo», обычно используется для обозначения программного обеспечения, программного кода, продукта, пакета офисных приложений. Когда говорится о проекте, в рамках которого ведется разработка этого офисного пакета, обычно употребляется название «OpenOffice.org», «проект OOo» или «проект разработки OOo». Следует также упомянуть, что существует еще один офисный пакет – StarOffice, в основе которого лежит тот же самый программный код[1], но он распространяется компанией Sun Microsystems как коммерческий продукт.[2]

Составляющие пакета OpenOffice

Один из наиболее существенных отличительных признаков OOo – тесная интеграция таких приложений, как текстовый процессор или электронная таблица, с другими компонентами пакета, что влечет за собой согласованность функциональных возможностей, единообразие меню и простоту использования. Компоненты, составляющие пакет OpenOffice, перечислены в табл. 8.1.

Таблица 8.1. Компоненты пакета OpenOffice

Название	Назначение	Пункт в меню File (Файл)→New (Создать)
OooWriter	Текстовый процессор	Text Document (Текстовый документ)
OooCalc	Электронная таблица	Spreadsheet (Электронную таблицу)
OooImpress	Редактор презентаций	Presentation (Презентацию)
OooDraw	Графический редактор	Drawing (Рисунок)
OOoHTML	Веб- (HTML-) редактор	HTML Document (Документ HTML)
OooMath	Редактор математических формул	Formula (Формулу)

[1] Точнее, фирма Sun Microsystems сама выступила инициатором создания проекта OpenOffice.org и опубликовала исходные тексты («blueprints») большей части кода очередной версии своего программного продукта StarOffice в 2000 году, которые и стали стартовой редакцией OpenOffice. Тем не менее проект StarOffice продолжает развиваться фирмой параллельно как коммерческий продукт для крупных корпоративных заказчиков. От OpenOffice его отличает главным образом более тщательное документирование и большие объемы встроенной справки. – *Примеч. науч. ред.*

[2] Свежие версии программного обеспечения OpenOffice, а также вся сопутствующая техническая документация могут быть свободно получены на ресурсе проекта *http://www.openoffice.org/* или на русскоязычном ресурсе *http://ru.openoffice.org/* (развитие и поддержку русскоязычной редакции OpenOffice выполняет компания «Инфра-Ресурс»: *http://www.i-rs.ru/about*). – *Примеч. науч. ред.*

В этой главе будет говориться о компонентах OOoWriter, OOoCalc и OOoImpress, остальные компоненты используются достаточно редко, а их функциональные возможности подробно описаны в книгах, названия которых упоминаются в тексте, и в электронных документах в Интернете.

Формат OpenDocument и OpenOffice2

Данный раздел книги описывает версию пакета OpenOffice 1.1, и потому будет представлять интерес для пользователей, работающих с версиями от 1.1.1 до 1.1.5. Однако к моменту публикации книги проект разработки OpenOffice.org выпустил пакет OpenOffice версии 2.[1]

В общем и целом OpenOffice 2 напоминает по внешнему виду современные версии Microsoft Office. Это должно упростить миграцию пользователей на открытый офисный пакет в Linux и других платформах.

Самое главное новшество, появившееся в версии 2, – новый формат файлов документов, получивший название «OASIS OpenDocument». Этот формат уже широко использовался отделами информационных технологий в научных и правительственных организациях (попробуйте для начала поискать в поисковой системе слова «Massachusetts» и «OpenDocument»).

Формат OpenDocument – это открытый формат XML, получивший в OpenOffice 2 новое расширение *.odt* в именах текстовых файлов, *.ods* – в именах файлов электронных таблиц и *.odp* – в именах файлов презентаций. (В первой версии используются расширения *.sxw*, *.sxc* и *.sxi* соответственно.) Формат OpenDocument по сути является обновленной версией формата XML, использовавшегося в версии 1, однако новый формат предлагает более широкие возможности, что делает его несовместимым с предыдущими версиями.

Соответственно, и программное обеспечение OpenOffice вплоть до версии 1.1.5 не может ни открывать, ни создавать файлы в формате OpenDocument, и потому не может работать с документами, созданными пользователями OpenOffice 2. Однако разработчиками из OpenOffice.org были добавлены в версию 1.1.5 дополнительные фильтры OpenDocument, чтобы пользователи версий 1.1 могли работать с прежним форматом документов и были в состоянии открывать документы в новом формате OpenDocument. Пользователям версии 1.1 или более ранних было бы желательно обновить свой пакет офисных приложений до OpenOffice 2, чтобы получить в свое распоряжение самые последние функциональные возможности и иметь возможность создавать документы OpenDocument.

OpenOffice Writer

OpenOffice Writer (известен также под названием OOoWriter) – это текстовый процессор, один из шести компонентов, составляющих пакет OpenOffice. В настоящее время OOoWriter по своему внешнему виду и функциональным возможностям очень напоминает Microsoft Word.

---

[1] На момент подготовки русскоязычного издания – версия 2.1. – *Примеч. науч. ред.*

Запуск OOoWriter

Настройка меню запуска приложений может сильно отличаться между разными дистрибутивами. Например, в Java Desktop System при выборе OOoWriter в меню запуска приложений открывается окно Templates and Documents (Шаблоны и документы) – New Document (Новый документ), где в левой панели следует выбрать пункт New Document (Новый документ), а затем в центральной панели – пункт Text Document (Текстовый документ).

Если на рабочем столе создать отдельный ярлык для программы OOoWriter, текстовый процессор можно будет запускать с помощью этого ярлыка. Создайте ярлыки на рабочем столе для наиболее часто используемых компонентов OpenOffice.

Кроме того, любой компонент можно запустить из любого другого запущенного модуля OpenOffice: в главном меню выберите пункт File (Файл)→New (Создать)→ [module] (<компонент>).

Открытие документов

Чтобы открыть документ OOoWriter или MS Word, можно перейти в каталог, где находится требуемый документ, и дважды щелкнуть мышкой на ярлыке документа или выбрать пункт меню File (Файл)→Open (Открыть). Затем, находясь в диалоге Open (Открыть), отыскать каталог с требуемым документом, выбрать нужный документ и щелкнуть на кнопке Open (Открыть).

Обратите внимание: документы MS Office (файлы с расширением *.doc*) открываются точно так же, как и документы OpenOffice. Документы MS Office могут быть сохранены как в оригинальном формате, так и в формате OpenOffice. В табл. 8.2 приводится полный список форматов, в которых можно сохранять документы.

Сохранение документов

После изменения документа выберите пункт меню File (Файл)→Save (Сохранить). По умолчанию новые файлы предлагается сохранить в каталоге */home/[user]/Documents*. Сохранить документ в текущем каталоге или в каталоге по умолчанию можно одним щелчком на кнопке Save (Сохранить) в панели инструментов.

Дополнительно о панелях инструментов будет рассказано в разделе «Идентификация панелей инструментов» далее в этой главе. Внешний вид панелей инструментов приводится на рис. 8.2.

Если необходимо сохранить документ в другом каталоге, изменить имя файла документа или его формат, следует выбрать пункт меню File (Файл)→Save As (Сохранить как). После этого на экране появится диалог Save As (Сохранить как), в котором можно определить необходимые параметры сохранения и щелкнуть на кнопке Save (Сохранить). Этот диалог будет описан в следующем разделе.

Сохранение документа в другом формате. Когда открывается существующий документ, по умолчанию он будет сохраняться в оригинальном формате. Чтобы сохранить документ в другом формате, необходимо выбрать пункт меню File (Файл) →Save As (Сохранить как), в открывшемся диалоге выбрать желаемый формат в раскрывающемся списке File Type (Тип файла) и щелкнуть на кнопке Save (Сохранить). Типы файлов, имеющиеся в раскрывающемся списке File Type (Тип файла), перечислены в табл. 8.2.

Таблица 8.2. Форматы и типы файлов, в которых могут быть сохранены документы

Формат файла	Расширение файла
OpenOffice 6.0/7 Текстовый документ	*.sxw*
OpenOffice 6.0/7 Шаблон текстового документа	*.stw*
MS Word 97/2000/XP	*.doc*
MS Word 95	*.doc*
MS Word 6.0	*Doc*
Adobe PDF	*.pdf*
Rich Text Format	*.rtf*
StarWriter 5.0	*.sdw*
Шаблон StarWriter 5.0	*.vor*
StarWriter 4.0	*.sdw*
Шаблон StarWriter 4.0	*.vor*
StarWriter 3.0	*.sdw*
Шаблон StarWriter 3.0	*.vor*
Текст	*.txt*
Кодированный текст	*.txt*
Документ HTML (OpenOffice Writer)	*.html; htm*
AportisDoc (Palm)	*.pdb*
DocBook	*.xml*
Pocket Word	*.psw*

Обратите внимание: доступны как «родной» формат текстовых документов OOo (с расширением *.sxw*), так и разные версии устаревших форматов MS Office наряду с прочими стандартами, включая формат веб-страниц – HTML.

При сохранении документа в форматах, отличных от формата текстовых документов OpenOffice, будет выведено предупреждение о возможной потере форматирования. Это происходит потому, что OOoWriter обладает особенностями, которые не поддерживаются другими текстовыми процессорами (и, соответственно, не могут быть сохранены в формате их документов). Таким образом, если необходима полная уверенность в сохранении форматирования, макросов и других элементов документа, всегда следует выбирать «родной» формат файла или хотя бы сохранять копию документа в «родном» формате.

Сохранение или экспорт документов в общераспространенные форматы. Пакет OpenOffice обладает возможностью сохранять документы в различных форматах, включая такие широко используемые, как PDF. Выбрав требуемый формат документа при сохранении, можно гарантировать, что он может быть открыт в других операционных системах, таких как Windows, Mac, Solaris и других.

Сохранение документов в форматах MS Office. Чтобы сохранить документ в формате MS Office, необходимо выбрать пункт меню File (Файл)→Save As (Со-

хранить как) и в диалоге Save As (Сохранить как) выбрать желаемую версию формата MS Office. В перечень поддерживаемых форматов входят форматы MSOffice следующих версий:

- Microsoft Word 97/2000/XP (*.doc*).
- Microsoft Word 95 (*.doc*).
- Microsoft Word 6.0 (*.doc*).

Экспорт и отправка документов по электронной почте

Иногда бывает необходимо быстро отправить документ в текущем его состоянии своему коллеге. OOo предлагает несколько пунктов меню, которые помогут присоединить текущий документ в требуемом формате к электронному письму.

Экспорт в формат Adobe PDF. Для этого необходимо щелкнуть на маленькой красной кнопке Export to PDF (Экспорт в PDF), расположенной в панели инструментов, в результате чего откроется диалог Export (Экспорт), где в поле File Type (Тип файла) уже будет выбран формат Adobe PDF. Обратите внимание: на рис. 8.1 можно заметить, что внешний вид диалога Export (Экспорт) практически не отличается от диалога Save As (Сохранить как).

Далее необходимо ввести имя файла, выбрать каталог для сохранения документа в формате PDF и щелкнуть на кнопке Save (Сохранить).

Того же результата можно добиться, если выбрать пункт меню File (Файл)→Export to PDF (Экспорт в PDF) и заполнить поля диалога Export (Экспорт), как это было только что описано.

PDF – это замечательный формат для пользователей GNU/Linux, и он достоин того, чтобы его использовали достаточно часто. В мире разнородных операцион-

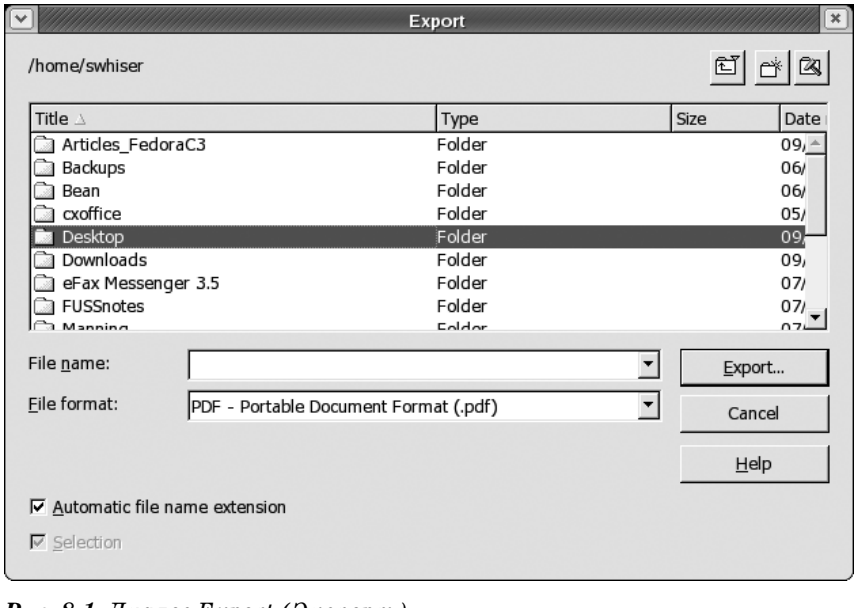


Рис. 8.1. *Диалог Export (Экспорт)*

ных систем PDF является наиболее универсальным из всех форматов и достаточно безопасным, поскольку значительно снижает вероятность случайного изменения содержимого документов.

Отправка документа по электронной почте. OOoWriter предоставляет возможность экспортировать документ в другой формат и отправить его по электронной почте всего лишь одним-двумя щелчками мышью. Чтобы отправить текущий открытый документ по электронной почте, необходимо выбрать пункт меню File (Файл)→Send (Отправить)→Document as Email (Документ как электронное письмо). Это приведет к запуску программы электронной почты, в которой будет открыто окно создания нового письма, а текущий документ уже будет вложен в письмо. После этого останется только заполнить поле адреса получателя, темы письма и, возможно, понадобится внести сопроводительный текст в тело письма, после чего можно будет щелкнуть на кнопке Send (Отправить).

Эта функция по умолчанию отправляет документы в открытом формате OpenOffice – XML (*.sxw*).

Отправка документа по электронной почте в формате PDF. Чтобы отправить текущий документ по электронной почте, необходимо выбрать пункт меню File (Файл) →Send (Отправить)→Document as PDF Attachment (Электронной почтой как PDF). В результате на экране появится диалог настройки параметров PDF, где будет предложено выбрать, пересылать ли диапазон страниц документа или весь документ целиком, а также указать степень сжатия файла. По умолчанию выбирается степень Print optimized (Для печати), чего вполне достаточно для большинства случаев.

Идентификация панелей инструментов

В OOoWriter имеются панели инструментов, к которым приходится обращаться достаточно часто: это панель главного меню, панель функций, контекстная панель и главная панель инструментов (рис. 8.2).

Это обычные панели инструментов, которые по умолчанию появляются в окне программы сразу же после установки. Для отображения других панелей инструментов требуются дополнительные настройки. Настраиваемые панели инструментов будут описаны в разделе «Настройка OOoWriter» далее в этой главе.

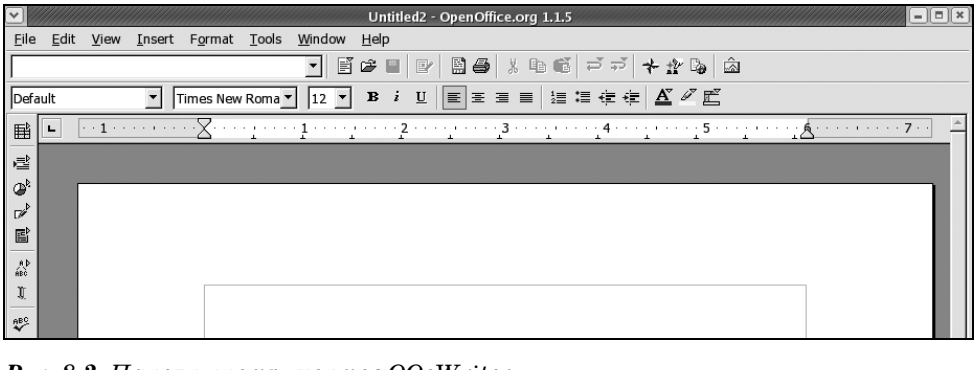


Рис. 8.2. *Панели инструментов OOoWriter*

Основы форматирования

В этом разделе описываются основные приемы форматирования простых документов.

Изменение форматирования символов одним щелчком. Кнопки форматирования текста, размещенные на контекстной панели и позволяющие изменять стиль начертания символов на жирный, курсивный и с подчеркиванием, не требуют подробного описания, поскольку назначение их инстинктивно понятно даже начинающим пользователям.

В дополнение к этим кнопкам многие пользуются быстрыми комбинациями клавиш Ctrl+B, Ctrl+I и Ctrl+U, которые производят те же самые изменения в выделенном тексте. Эти изменения отражаются на отдельных словах, находящихся под текстовым курсором, поэтому, если нужно изменить стиль начертания отдельного слова, не обязательно выделять его (перемещая текстовый курсор с нажатой клавишей Shift или с помощью мыши).

Форматирование символов, параграфов и страниц. Более сложное форматирование текста, абзацев или даже целых страниц выполняется с помощью меню Format (Формат). После выбора пункта меню Format (Формат)→Character (Символы), Format (Формат)→Paragraph (Абзац) или Format (Формат)→Page (Страница) откроется диалог Character (Символы), Paragraph (Абзац) или Page Style: Default (Стиль страницы: Обычный), соответственно.

Вставка колонтитулов. Чтобы добавить верхний колонтитул, в главном меню необходимо выбрать пункт Insert (Вставка)→Header (Верхний колонтитул), а затем в раскрывающемся меню выбрать пункт Default (Обычный). В результате в документе появится рамка верхнего колонтитула. Сюда можно ввести текст, который появится в верхней части каждой страницы документа.

Добавление нижнего колонтитула производится аналогичным образом. Только на этот раз необходимо выбрать пункт меню Insert (Вставка)→Footers (Нижний колонтитул) и выбрать Default (Обычный).

Порядок изменения верхнего и нижнего колонтитулов в середине документа описывается в разделе «Изменение стиля в середине документа» далее в этой главе.

Нумерация страниц. Для большинства документов желательно, чтобы на каждой странице появлялись порядковые номера страниц в верхнем или нижнем колонтитуле. Чтобы нумерация страниц документа производилась автоматически, необходимо добавить верхний или нижний колонтитул (в зависимости от того, где должны располагаться номера страниц) и щелчком мыши на колонтитуле установить текстовый курсор. Затем перейти в главное меню и выбрать пункт Insert (Вставка)→Fields (Поля). При этом появится дополнительное раскрывающееся меню с пунктами: Date (Дата), Time (Время), Page Number (Номер страницы), Page Count (Количество страниц), Subject (Тема), Title (Заголовок), Author (Автор) и Other (Дополнительно).

Если выбрать пункт Page Number (Номер страницы), на каждой странице в позиции текстового курсора появится порядковый номер. Если необходимо, чтобы номера страниц располагались по правому краю, достаточно после вставки номеров просто щелкнуть на кнопке выравнивания Align Right (По правому краю) в контекстной панели.

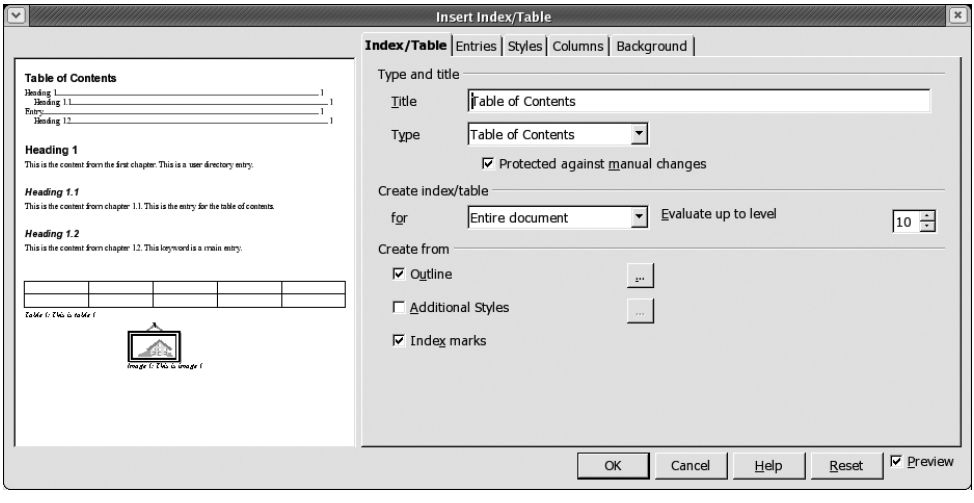


Рис. 8.3. *Диалог Insert Index/Table (Вставить оглавление/указатель)*

Изменение порядка нумерации страниц в отдельных частях документа описывается в разделе «Изменение стиля в середине документа» далее в этой главе.

Создание оглавления. При работе над большими документами очень удобно пользоваться способностью OOoWriter автоматически создавать оглавление документа. Эта возможность пользуется большой популярностью, поскольку создание оглавлений и алфавитных указателей вручную, особенно для больших документов, отнимает слишком много времени.

Чтобы создать оглавление документа, в котором будут перечислены все заголовки документа, необходимо выбрать пункт меню Insert (Вставка)→Indexes and Tables (Оглавление и указатели), а затем из раскрывающегося меню выбрать пункт Indexes and Tables (Оглавление и указатели). Затем, щелкнув на кнопке OK в диалоге Insert Index/Table (Вставить оглавление/указатель) (рис. 8.3), можно будет вставить типичное оглавление.

Диалог позволяет выбирать тип оглавления или указателя из числа имеющихся: Table of Contents (Оглавление), Alphabetical Index (Алфавитный указатель), Illustration Index (Список иллюстраций), Index of Tables (Список таблиц), User-Defined (Определяемый пользователем), Table of Objects (Таблица объектов) и Bibliography (Библиография). В этом же диалоге можно определить форматирование указателей и оглавлений, их местоположение, максимальный уровень вложенности заголовков и другие параметры.

Печать документа

Чтобы напечатать текущий документ на принтере, достаточно щелкнуть на кнопке с изображением принтера в панели функций.

Если перед печатью документа необходимо выполнить дополнительные настройки параметров печати, необходимо выбрать пункт меню File (Файл)→Print (Печать) или нажать комбинацию клавиш Ctrl+P. В появившемся диалоге можно

будет выбрать принтер, который не используется по умолчанию (если таковой имеется), ограничить диапазон печатаемых страниц или задать число печатаемых копий документа. Здесь же можно выбрать функцию печати в файл.

Дополнительные возможности форматирования

В следующих разделах описываются более сложные приемы форматирования, однако некоторые из возможностей выходят за рамки этой главы. Поэтому в данном разделе будут упомянуты некоторые из особенностей, которые мы не способны охватить в одной главе.

В объемных документах часто используются такого рода элементы, как рамки, врезки и разделы. Это позволяет отдельно форматировать врезки, цитаты или выделять элементы, которые должны отличаться от обычного текста. Они предоставляют такие возможности, как добавление окрашенного или заштрихованного фона, изменение шрифта и оформление текста в виде нескольких колонок. Текст, содержащийся в подобных элементах, может даже располагаться в нескольких из них, разбросанных по всему документу. Это особенно востребовано, например, при создании информационных бюллетеней для придания им более привлекательного внешнего вида.

Шаблоны

В состав OOoWriter включены самые разнообразные шаблоны форматирования и средства для их создания, редактирования, импортирования и управления ими. Чтобы получить доступ к шаблонам, необходимо выбрать пункт меню File (Файл)→New (Создать)→Templates and Documents (Шаблоны и документы), в результате на экране появится диалог Templates and Documents (Шаблоны и документы). В левой панели диалога следует выбрать ярлык Templates (Шаблоны), как показано на рис. 8.4.

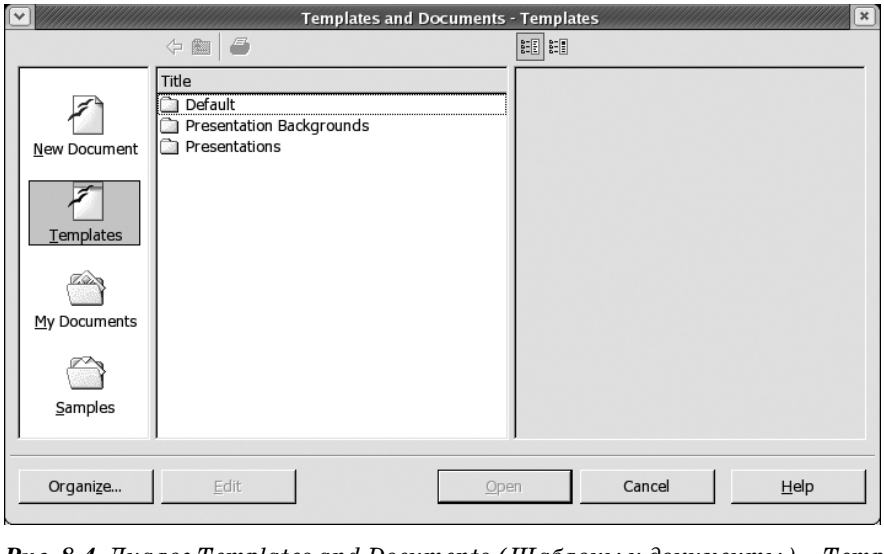


Рис. 8.4. *Диалог Templates and Documents (Шаблоны и документы) – Templates (Шаблоны)*

Здесь можно выбрать требуемый шаблон, открыть его, отредактировать и сохранить как обычный документ. Однако документы, созданные таким образом, не будут связаны с файлом шаблона, из которого они были получены. Дополнительная информация по этой теме приводится в разделе «Связывание с файлом шаблона» далее в этой главе.

Сохранение собственного документа в виде шаблона. Любой документ, созданный в пределах файловой системы, может исполнять роль шаблона. Достаточно часто пользователи используют существующие документы в качестве шаблонов деловых писем, факсов, заметок, заменяя лишь несколько слов при создании нового документа из шаблона. Такая практика получила широкое распространение и хорошо зарекомендовала себя, однако пользователи могли бы избежать лишних усилий, если бы взяли на вооружение те преимущества, которые дает им возможность управления шаблонами, имеющаяся в OOoWriter, и особенно его возможность связывания шаблонов.

Создание нового шаблона. Чтобы создать новый шаблон, необходимо открыть новый текстовый документ (или использовать существующий) и выполнить форматирование документа, которое нужно для шаблона, а затем выбрать пункт главного меню File (Файл)→Templates (Шаблоны)→Save (Сохранить). В результате на экране появится диалог Templates (Шаблоны), в котором можно будет указать название шаблона и выбрать категорию, в которой этот шаблон будет сохранен. Таким же способом можно создать произвольное число шаблонов.

Файлы, сохраненные как шаблоны, автоматически получают расширение *.stw*.

Редактирование шаблонов. Шаблоны можно редактировать или просто просматривать точно так же, как любой другой документ, однако редактировать шаблоны рекомендуется с особой осторожностью, потому что очень просто открыть файл шаблона и затем сохранить его по ошибке как обычный файл *.sxw*, что может вызвать ошибки связывания с шаблоном и местом его расположения.

Чтобы отредактировать шаблон, необходимо выбрать пункт меню File (Файл)→New (Создать)→Templates and Documents (Шаблоны и документы). В результате будет открыт диалог Templates and Documents (Шаблоны и документы) в папке с шаблонами по умолчанию. Далее следует щелкнуть на папке Templates (Шаблоны) и отыскать шаблон, который необходимо отредактировать. После этого нужно щелкнуть мышкой на шаблоне, чтобы выделить его. Активизируется кнопка Edit (Редактировать), расположенная в нижней части диалога, и щелчком на ней можно будет открыть шаблон для редактирования. Когда шаблон открывается для редактирования таким способом, каталог сохранения и формат файла в диалоге Save (Сохранить) подставляются автоматически, что существенно снижает вероятность сохранить шаблон не в том формате или не в том месте.

Управление шаблонами. Любой документ OOoWriter можно сохранить в виде шаблона или позднее переместить его в одну из папок в коллекции шаблонов с помощью менеджера шаблонов (рис. 8.5). Вызвать диалог менеджера шаблонов можно с помощью пункта главного меню File (Файл)→Templates (Шаблоны)→Organize (Управление).

В правой панели диалога можно искать нужные документы и перетаскивать их в папки, расположенные в панели шаблонов с левой стороны.

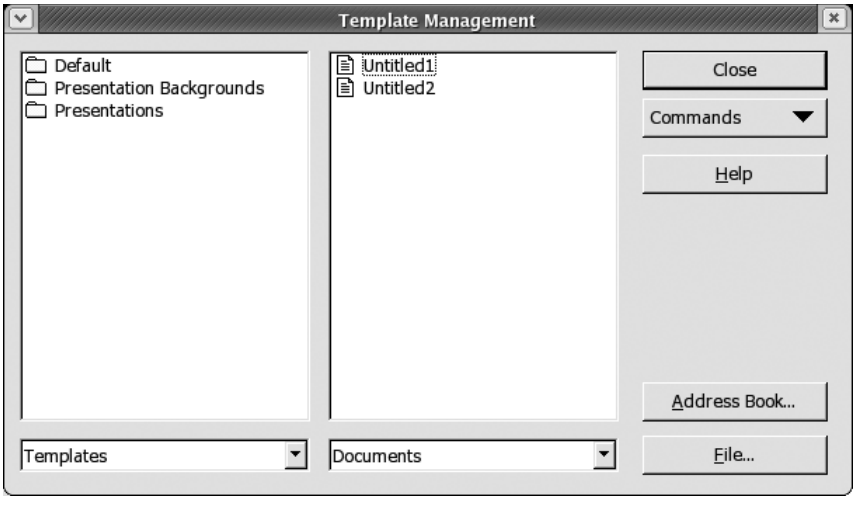


Рис. 8.5. *Диалог Template Management (Управление шаблонами)*

Кроме того, менеджер шаблонов предоставляет возможность импорта, обновления и изменения параметров печати, ассоциированных с шаблонами.

Импорт шаблонов. Файлы шаблонов из папки по умолчанию фактически хранятся в каталоге */home/[user]/OpenOffice.org/user/template*. Другие шаблоны, которые можно увидеть в диалоге Template Management (Управление шаблонами), фактически хранятся в каталоге */home/[user]/OpenOffice.org/share/template/english*. (Это позволяет каждому пользователю в многопользовательской системе определять свои собственные шаблоны, не оказывая влияния на других пользователей.)

Чтобы импортировать шаблоны из MS Word или другого источника (включая коллекции шаблонов, которые можно найти в Интернете), можно вручную скопировать файлы шаблонов в указанные каталоги, после чего они появятся в соответствующих папках диалога Template Management (Управление шаблонами). Шаблоны, скопированные таким образом, будут доступны в мастере создания новых документов из шаблонов.

Можно также воспользоваться функцией импорта, имеющейся в менеджере шаблонов и предназначенной для размещения внешних шаблонов в нужном каталоге и в нужном формате (*.stw*). Тем самым гарантируется, что связь между шаблонами и документами, созданными на их основе, не будет нарушена (подробнее об этом рассказывается в следующем разделе).

Третий способ импортирования шаблонов заключается в том, чтобы выбрать пункт File (Файл)→Save As (Сохранить как), выбрать тип файла «OpenOffice 6.0/7 Шаблон текстового документа (.stw)» и указать в качестве каталога сохранения один из двух упомянутых выше каталогов.

Связывание с файлом шаблона. Документы, созданные на основе шаблонов, связываются с файлами шаблонов. Эту связь можно представить себе как взаимоотношения «родителя» (файл шаблона) и «потомка» (файл документа, созданного на основе шаблона). Связь «родитель – потомок» – это одно из основных

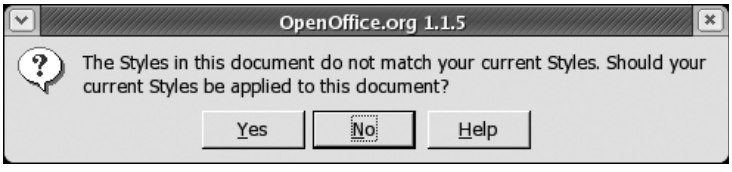


Рис. 8.6. *Запрос согласия на принятие изменений из шаблона*

преимуществ, которые дает использование шаблонов. Например, как много документов ни накапливалось бы в каталоге *Documents*, в любой момент можно изменить их форматирование буквально одним движением, то есть редактированием единственного файла шаблона. Если файл шаблона подвергался изменениям, каждый раз, когда будет открываться дочерний документ, будет появляться запрос о том, стоит ли принять изменения, которые были обнаружены в файле шаблона, как показано на рис. 8.6.

Однако связь между документом и файлом шаблона будет нарушена, если позднее файл шаблона будет сохранен выбором пункта меню File (Файл)→Save As (Сохранить как) или щелчком по кнопке Save (Сохранить) в контекстной панели. Поэтому для сохранения шаблонов всегда нужно использовать пункт меню File (Файл)→Templates (Шаблоны)→Save (Сохранить), если необходимо сохранить связь между шаблоном и созданными на его основе документами.

Изменение шаблона по умолчанию для всех новых документов. Как уже упоминалось ранее, стандартный пустой документ создается с помощью пункта меню File (Файл)→New (Создать)→Text Document (Текстовый документ) на основе шаблона Default (Обычный), который присутствует в диалоге Templates and Documents – New Document (Шаблоны и документы – Новый документ) (рис. 8.7).

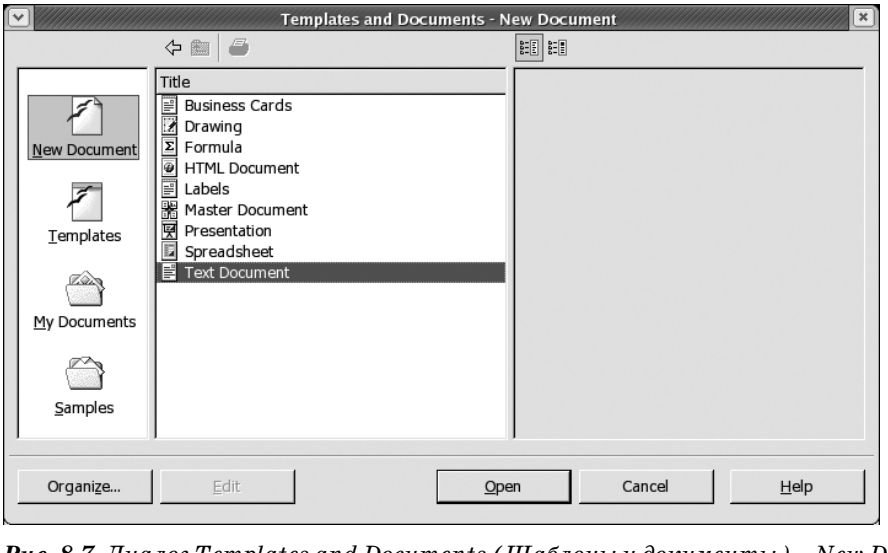


Рис. 8.7. *Диалог Templates and Documents (Шаблоны и документы) – New Document (Новый документ)*

Чтобы изменить шаблон, используемый по умолчанию при создании новых текстовых документов, прежде всего необходимо создать новый шаблон с соответствующим форматированием (куда при желании можно добавить свои собственные стили), как описывалось в разделе «Создание нового шаблона». Далее нужно сохранить шаблон: выбрать пункт меню File (Файл)→Templates (Шаблоны)→Save (Сохранить), ввести имя файла с шаблоном (пусть это будет имя *newdefault*) и щелкнуть в левой панели на папке с названием категории Default (Обычный), чтобы сохранение было выполнено в эту папку.

После этого нужно открыть диалог Template Management (Управление шаблонами), выбрав пункт меню File (Файл)→Templates (Шаблоны)→Organize (Управление), и дважды щелкнуть в левой панели на папке Default (Обычный), где можно будет увидеть шаблон *newdefault*. Затем нужно щелкнуть на шаблоне *newdefault*, чтобы выделить его, потом на командной кнопке, расположенной в правой части диалога, чтобы добраться до раскрывающегося меню и выбрать в нем пункт Set As Default Template (Сделать шаблоном по умолчанию).

Чтобы восстановить оригинальный шаблон по умолчанию, достаточно просто еще раз щелкнуть на командной кнопке и выбрать пункт меню Reset Default Template (Восстановить стандартный шаблон)→Text Document (Текстовый документ).

Мастер быстрого создания документов из шаблонов. Мастер напоминает шаблоны на стероидах («усиленные» шаблоны). Он позволяет создавать документы на основе шаблонов, но при этом требует выполнения пошаговой процедуры настройки документа, перед тем как тот будет открыт. Таким образом, мастер создания документов из шаблонов может рассматриваться как инструмент для начинающих пользователей, желающих быстро научиться работать с OOoWriter.

Доступ к мастеру осуществляется через пункт меню File (Файл)→AutoPilot (Мастер), после выбора которого раскрывается дополнительное меню, как показано на рис. 8.8.

Мастер выполняет пошаговую процедуру создания документов из шаблонов, в перечень которых входят письма, факсы, повестки дня, презентации, веб-страницы, формы и отчеты.

Кроме того, в мастере имеются несколько утилит управления документами или преобразования их содержимого: Document Converter (Конвертер документов), Euro Converter (Конвертер Евро), StarOffice 5.2 Database Import (Импорт базы данных StarOffice 5.2) и Address Data Source (Источники данных адресов).

Стили

При работе в составе коллектива бывает необходимо обеспечить единообразное форматирование всех документов. В этом случае на помощь придут стили. Другими словами, любое форматирование текста можно быстро преобразовать в стиль, а затем применять его повсюду парой щелчков.

На рис. 8.9 показана кнопка на панели функций (третья справа, выделенная), она быстро открывает окно со списком стилей, которое позволяет манипулировать ими. Кроме того, окно со списком стилей открывается нажатием клавиши F11.

В окне Stylist (Стилист) можно выбрать одну из пяти категорий стилей: стили абзаца, стили символов, стили рамок, стили страниц и стили списков. Чтобы пе-

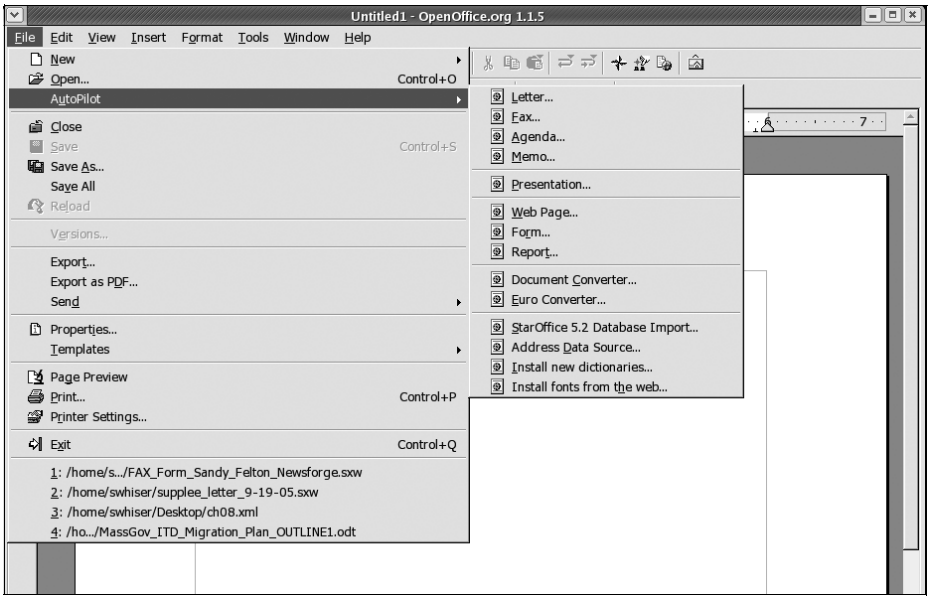


Рис. 8.8. *Мастер создания документов из шаблонов*

рейти из одной категории в другую, достаточно просто щелкнуть на соответствующей кнопке в панели инструментов Стилиста.

Стилист. Интерфейс доступа к стилям в OOoWriter реализован в виде плавающей палитры, которая называется Stylist (Стилист). Она вызывается нажатием клавиши F11 или щелчком на кнопке Stylist (Стилист) в панели функций. На кнопке вызова Стилиста изображена страница с маленькой рукой в левом нижнем углу. По умолчанию во время открытия Стилист активизирует список стилей абзаца и выбирает режим Automatic (Автоматически), как показано на рис. 8.10.

Щелкая на кнопках в панели инструментов Стилиста, можно быстро ознакомиться с полным списком стилей, которые включены в состав OOoWriter по умолчанию.

Применение стилей. Чтобы применить стиль символов, нужно щелкнуть на кнопке выбора категории символьных стилей в панели инструментов окна Стилиста (вторая слева, с изображением символа A). В результате станут доступными все стили символов (Стилист автоматически перейдет в режим отображения All (Все)).

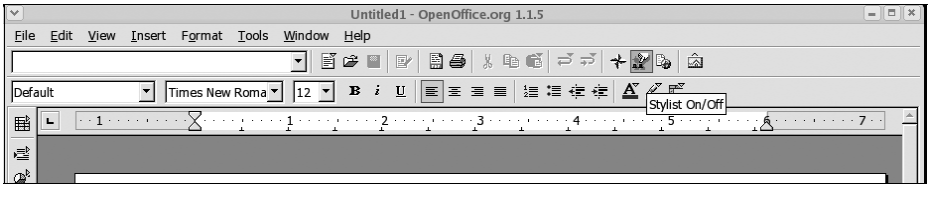


Рис. 8.9. *Кнопка включения-выключения окна Стилиста*

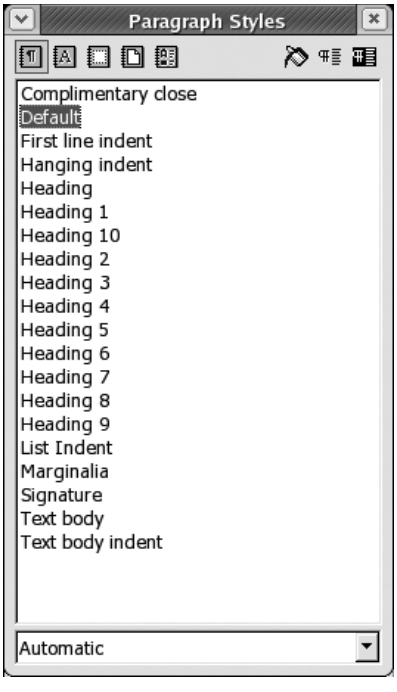


Рис. 8.10. *Стилист с открытым списком стилей абзаца*

Чтобы применить, например, стиль курсивного начертания, нужно щелчком мыши выделить стиль Emphasis (Выделение) (по умолчанию пятый от начала списка), а затем щелкнуть на кнопке применения стиля, где изображено ведерко с краской, третья справа на панели инструментов (рис. 8.11).

После щелчка на кнопке применения стиля указатель мыши превратится в небольшое ведерко с краской. После этого нужно щелкнуть на выбранном слове или выделить мышью участок текста. Таким образом можно применить выбранный стиль к любым частям текста, к которым прикоснется указатель мыши. Закрыть окно Стилиста можно нажатием клавиши F11, щелчком на кнопке с крестиком в его правом верхнем углу или щелчком на кнопке вызова окна Стилиста в панели функций OOoWriter.

Изменение стилей. Чтобы приступить к изменению стилей, нажмите комбинацию Ctrl+F11. В результате на экране появится диалог Style Catalog (Каталог стилей), показанный на рис. 8.12. Каталог стилей также можно вызвать из главного меню, выбрав пункт Format (Формат)→Styles (Стили)→Catalog (Каталог).

Ранее уже демонстрировалось изменение определенного абзаца или набора символов. То же самое можно делать и со стилями. Например, если необходимо, чтобы элементы списка выравнивались иначе, чем определено по умолчанию в стиле списка, можно отредактировать стиль списка и тем самым изменить порядок выравнивания во всех списках сразу. При изменении какого-либо стиля немедленно изменяются все существующие элементы в документе, оформленные этим сти-

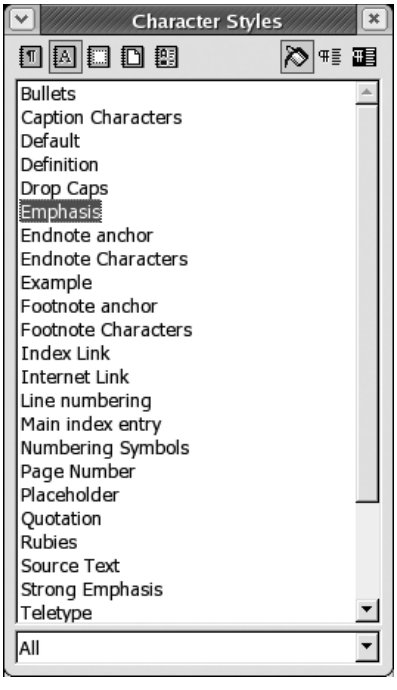


Рис. 8.11. *Стилист готов применить выбранный стиль*

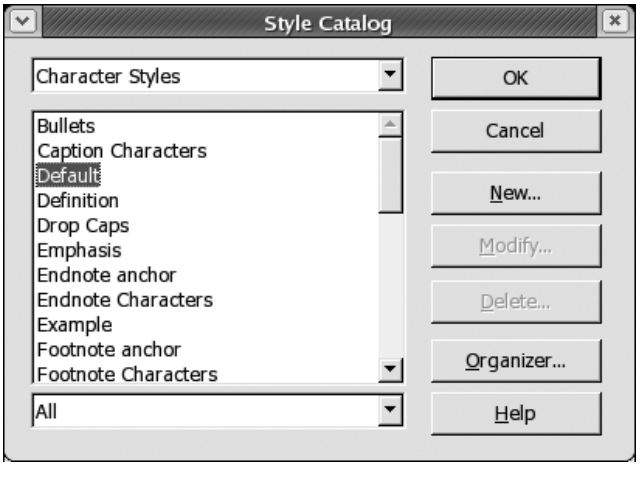


Рис. 8.12. *Диалог Style Catalog (Каталог стилей)*

лем, а также элементы этого стиля, которые будут создаваться позднее. В этом разделе рассказывается, как изменить стиль, а в следующем – как создать полностью новый стиль, благодаря чему вы получите возможность творить такое, о чем создатели OOoWriter и не подозревали.

Возможность быстрого изменения стилей и их использование вместо непосредственного форматирования текста относятся к основным составляющим повышения производительности труда. С помощью стилей возможно эффективное форматирование больших документов, которым наверняка многие захотят воспользоваться.

Текущий стиль текста в позиции текстового курсора автоматически выбирается в диалоге Style Catalog (Каталог стилей). Это очень удобно на тот случай, если у пользователя возникнет желание изменить определенный стиль во всем документе: достаточно будет поместить текстовый курсор в текст с нужным стилем и изменить форматирование текста.

В окне диалога Style Catalog (Каталог стилей) нужно выделить стиль, подлежащий изменению, и щелкнуть на кнопке Modify (Изменить), расположенной в правой части окна диалога. В результате откроется диалог настройки стиля (на рис. 8.12 изменяется стиль Default (Обычный)). Внешний вид диалога настройки стиля показан на рис. 8.13. Здесь можно изменять любые параметры, доступные для модификации.

Другой способ изменить стиль: щелкнуть правой кнопкой мыши на названии стиля в окне Стилиста и выбрать в контекстном меню пункт New (Создать), Modify (Изменить) или Delete (Удалить). В случае выбора пункта Modify (Изменить) откроется диалог настройки стиля, где можно будет выполнить необходимые изменения.

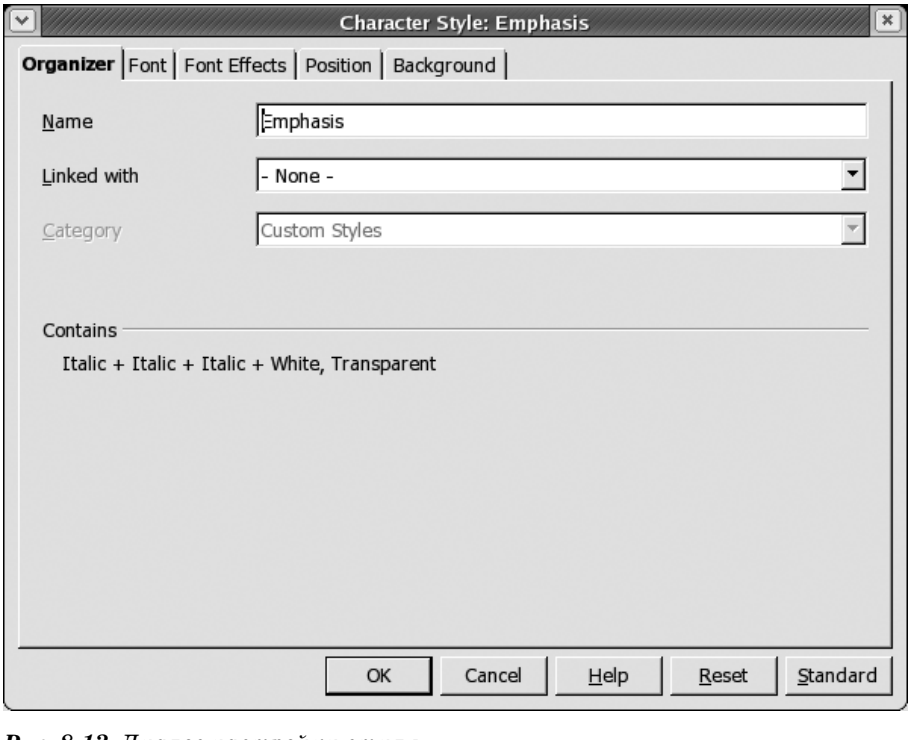


Рис. 8.13. *Диалог настройки стиля*

Обновление стилей. Помимо создания нового стиля «с нуля» есть возможность быстро изменить существующий стиль, применив формат выделенной последовательности символов, абзаца или страницы.

Чтобы обновить какой-либо стиль, нужно открыть окно Стилиста нажатием клавиши F11. Далее выберите категорию, в который находится требуемый стиль: абзац, символ или страница. Затем щелкните на тексте, стиль которого нужно скопировать или обновить. Например, можно «позаимствовать» форматирование абзаца, которое раньше было выполнено вручную. После этого в окне Стилиста выберите стиль, который подлежит обновлению, и наконец щелкните на кнопке Update Style (Обновить стиль), крайней справа в панели инструментов Стилиста.

Добавление (или создание) новых стилей. Текстовый процессор OOoWriter поставляется с массой предопределенных стилей, но несмотря на это неизбежно будет возникать необходимость в добавлении *новых* стилей. Эти стили также называют *заказными стилями*; они включаются в состав документа, в котором были созданы, когда документ сохраняется.

Чтобы добавить новый стиль, сначала нужно открыть окно Стилиста нажатием клавиши F11. Затем выберите категорию и существующий стиль в списке, на основе которого предполагается создать новый стиль. Щелкните правой кнопкой мыши на выбранном стиле и в контекстном меню выберите пункт New (Создать). В результате на экране появится диалог настройки стиля (см. рис. 8.13). В нем вы можете определить параметры нового стиля, включая категорию.

Существует два альтернативных способа добавления нового стиля. Первый состоит в том, чтобы щелкнуть на кнопке New Style from Selection (Создать стиль из выделенного), второй справа в панели инструментов Стилиста. В результате на экране появится диалог Create Style (Создать стиль), где можно выбрать новый стиль из предлагаемого списка или ввести название нового стиля, как показано на рис. 8.14.

Пожалуй, самый лучший способ создать новый стиль, который близко не напоминает ни один из существующих, состоит в том, чтобы открыть диалог Style Catalog (Каталог стилей) нажатием комбинации Ctrl+F11 и щелкнуть на кнопке

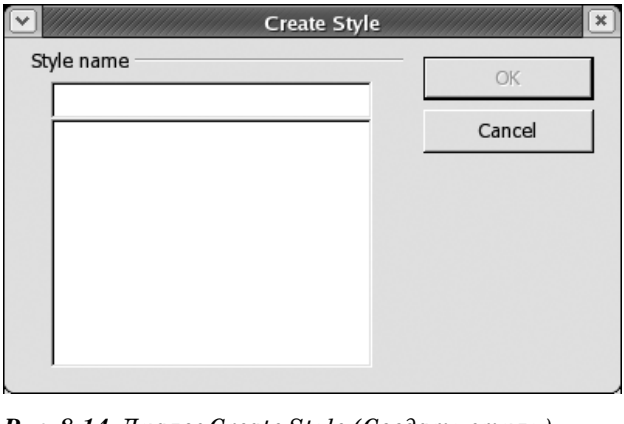


Рис. 8.14. *Диалог Create Style (Создать стиль)*

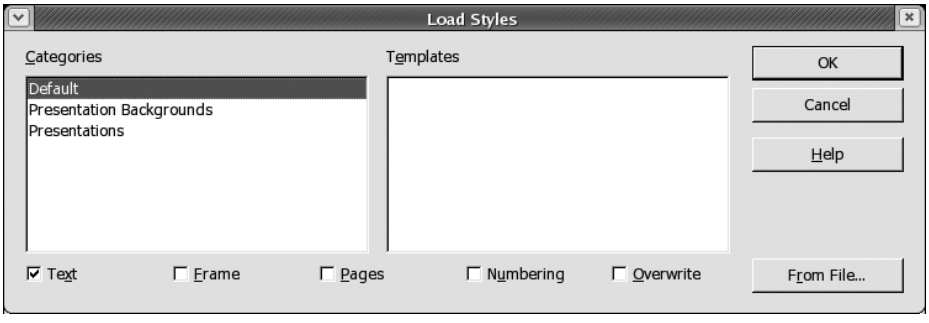


Рис. 8.15. *Диалог Styles Load (Загрузить стили)*

New (Создать). В результате будет открыт диалог настройки стиля, где вы можете выполнить настройку всех параметров нового стиля.

Изменение стиля в середине документа. Чтобы изменить стиль страницы, колонтитулов или начать новый порядок нумерации страниц в середине документа, обычно требуется вручную вставить разрыв – установить текстовый курсор в точку разрыва и выбрать пункт главного меню Insert (Вставка)→Break (Разрыв). В появившемся диалоге Insert Break (Вставить разрыв) будет предоставлена возможность выбрать новый стиль страницы или изменить порядок нумерации страниц. Чтобы изменить колонтитулы, сначала нужно создать новый стиль страницы с соответствующими колонтитулами, а затем использовать этот стиль при вставке разрыва.

Загрузка (передача) стилей. Существует возможность загружать стили в текущий документ из другого документа или шаблона. Для этого нужно открыть диалог Load Styles (Загрузить стили) (рис. 8.15), выбрав пункт главного меню Format (Формат)→Styles (Стили)→Load (Загрузить). Здесь можно выбрать файл, содержащий требуемые стили, и загрузить любые из них, устанавливая соответствующие флажки в нижней части окна.

Совместная работа над документами

Когда над одним документом работают несколько человек, передавая друг другу черновой вариант, очень удобно использовать функцию отслеживания изменений. Она позволяет выделить цветом все изменения в документе, где работа каждого пользователя выделяется своим цветом.

Включение функции отслеживания изменений. Чтобы включить функцию отслеживания изменений, необходимо выбрать пункт меню Edit (Правка)→Changes (Изменения), а затем щелкнуть на пунктах Record (Запись) и Show (Показать). После этого данные параметры сохранятся вместе с документом и будут действовать до тех пор, пока кто-либо не сохранит документ с выключенными параметрами.

Сравнение документов. Чтобы сравнить два документа, нужно открыть первый документ и выбрать пункт главного меню Edit (Правка)→Compare Documents (Сравнить документы). В результате на экране появится диалог Insert (Вставка), где будет предоставлена возможность выбрать или ввести имя файла второго документа. После щелчка на кнопке Insert (Вставить) процедура вставки объединит

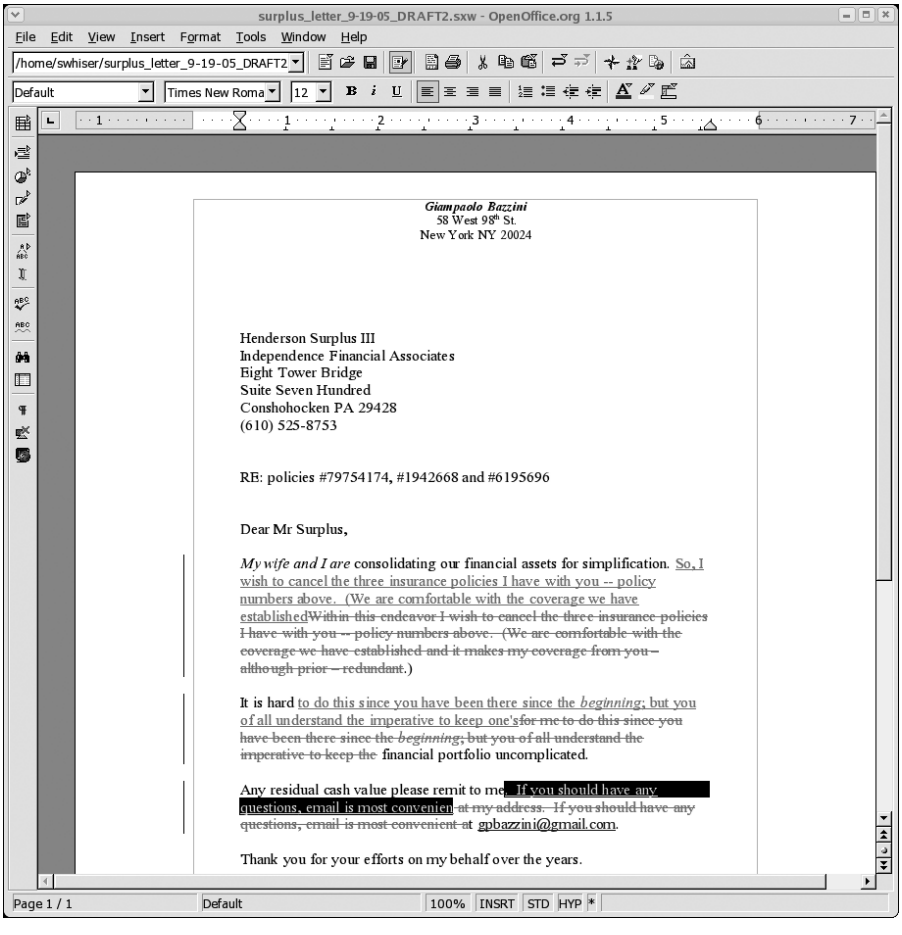


Рис. 8.16. *Пример сравнения двух документов*

два документа и покажет результаты с помощью функции отслеживания изменений таким образом, как будто во второй документ вносились изменения, необходимые для того, чтобы получить первый документ. Пример сравнения двух документов приводится на рис. 8.16.

Контроль версий. Возможность осуществления контроля версий документов, предусмотренная в OOoWriter, позволяет хранить различные версии документа в одном-единственном файле. Это позволяет сэкономить дисковое пространство и обеспечить быстрый доступ к старым версиям документа. Таким образом, если в документ были внесены изменения, которые позднее окажутся нежелательными, можно будет просто отменить их. Если кого-то заинтересует время, когда были внесены те или иные изменения, можно будет просто просмотреть более ранние версии документа.

Чтобы получить доступ к механизму контроля версий, необходимо выбрать пункт главного меню File (Файл)→Versions (Версии). После этого на экране по-

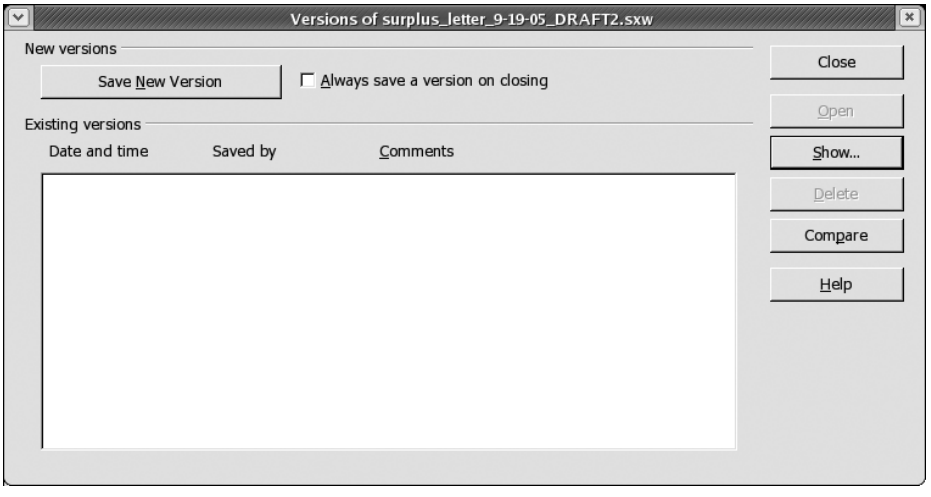


Рис. 8.17. *Диалог Versions (Версии)*

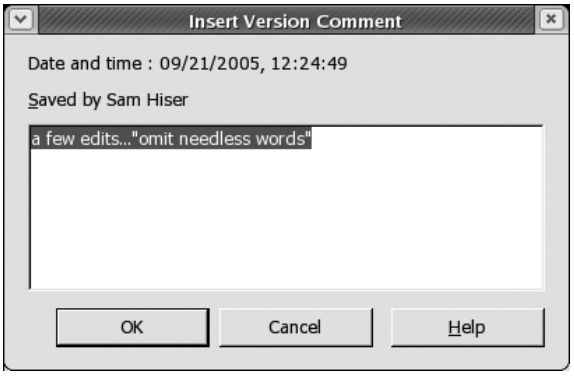


Рис. 8.18. *Диалог Insert Version Comment (Вставить комментарий к версии)*

явится диалог Versions (Версии) (рис. 8.17). Чтобы сохранить новую версию текущего документа, следует выбрать пункт главного меню File (Файл)→Versions (Версии) и щелкнуть на кнопке Save New Version (Сохранить новую версию), расположенной в верхнем левом углу диалога. После этого появится диалог Insert Version Comment (Вставить комментарий к версии) (рис. 8.18), где можно ввести несколько фраз, описывающих внесенные изменения и их причину. Кроме того, документирование изменений позволит в будущем отличать версии между собой без необходимости открывать их.

Если производить сохранение версии, выбирая пункт меню File (Файл)→Save As (Сохранить как), никакого упоминания о предыдущих версиях сохранено не будет, вместо этого будет создан совершенно новый документ. Разумеется, контроль версий можно начать в новом документе сначала, используя этот документ в качестве основы.

Чтобы открыть определенную версию документа, имеющуюся в списке диалога Versions (Версии), нужно выбрать пункт главного меню File (Файл)→Versions (Версии), выделить нужную версию и щелкнуть на кнопке Open (Открыть). В результате будет открыта требуемая версия в режиме «только для чтения». Эту версию можно будет сохранить в виде самостоятельного документа без информации о других версиях, для чего достаточно выбрать пункт главного меню File (Файл)→Save As (Сохранить как).

Чтобы просмотреть различия между версиями, нужно в диалоге Versions (Версии) щелкнуть на кнопке Compare (Сравнить). В результате будут показаны все изменения в документе (как если бы была выбрана операция сравнения двух документов) и будет предоставлена возможность принять или отклонить каждое из изменений.

Навигатор

Навигатор – это плавающая панель, аналогичная панели Стилиста, с помощью которой можно быстро перемещаться по всему документу. Вызов панели Навигатора производится щелчком на кнопке Navigator (Навигатор), расположенной в панели инструментов левее кнопки вызова Стилиста, или нажатием функциональной клавиши F5.

В панели Навигатора отображается схема документа, содержащая все его элементы и позволяющая быстро перемещаться к любому из них. В число таких элементов включаются следующие категории: заголовки, таблицы, врезки, графические объекты, объекты OLE, закладки, разделы, гиперссылки, ссылки, указатели, примечания и рисованные объекты. Если в окне Навигатора щелкнуть на знаке «плюс» рядом с выбранной категорией, можно будет выбрать нужное из списка элементов и сразу же переместиться в то место документа, где этот объект находится.

Быстрые комбинации клавиш

В этом разделе перечислены наиболее часто используемые комбинации клавиш, позволяющие ускорить работу над документом. Быстрые комбинации клавиш работают быстрее, чем выбор пункта меню мышью, потому что для этого не нужно убирать руки с клавиатуры. Те, кто опасается заработать синдром переутомления от чрезмерного использования мыши, найдут эти комбинации весьма удобными.

Настройка быстрых комбинаций клавиш. Привязки комбинаций клавиш, приведенные в табл. 8.3, – это всего лишь параметры настройки по умолчанию. Пользователи и администраторы системы могут свободно изменять их исходя из своих собственных предпочтений и привычек, для чего нужно выбрать пункт меню Tools (Сервис)→Configure (Настройка)→Keyboard (Клавиатура).

Настройка значений по умолчанию для функциональных клавиш также может оказать помощь в процессе перемещения по рабочему столу. OpenOffice предлагает возможность настройки комбинаций четырех режимов: F[1–12], Shift+F[1–12], Ctrl+F[1–12] и Shift+Ctrl+F[1–12], что открывает широкий простор в настройке привязок функциональных клавиш, которые могут помочь повысить производительность труда.

Таблица 8.3. Общераспространенные комбинации клавиш, позволяющие избежать использования мыши

Функция	Комбинация
Копировать текст	Ctrl+C
Вырезать текст	Ctrl+X
Вставить текст	Ctrl+V
Жирный шрифт	Ctrl+B
Курсивный шрифт	Ctrl+I
Шрифт с подчеркиванием	Ctrl+U

Поиск в документе в режиме «Найти и заменить»

Чтобы отыскать и заменить определенные последовательности символов в документе, нужно открыть диалог Find & Replace (Найти и заменить) нажатием комбинации Ctrl+F. Доступ к этому диалогу можно получить с помощью пункта меню Edit (Правка)→Find & Replace (Найти и заменить).

Введите слово или последовательность символов, которую нужно найти, в поле Search For (Найти) (в верхней части диалога) и, если ее нужно заменить, введите последовательность символов в поле Replace With (Заменить на). Запустите процесс поиска щелчком на кнопке Find (Найти). Процедура поиска отыщет последовательность символов, ближайшую к текстовому курсору. После этого можно щелкнуть на кнопке Replace (Заменить), если в этом возникнет необходимость. Если вы не желаете производить замену, просто щелкните на кнопке Find (Найти), чтобы отыскать следующее совпадение.

Вставка гиперссылок

Вставка гиперссылок – текстовых ссылок на URL в Web – в документы давно стала обычным делом. Чтобы вставить ссылку, необходимо выбрать пункт главного меню Insert (Вставка)→Hyperlink (Гиперссылка). В результате на экране появится диалог Hyperlink (Гиперссылка), где можно ввести имя ссылки, саму ссылку (начинающуюся с *http://*) в поле Target (Адрес) и текст ссылки, который будет указан в документе во втором снизу поле Text (Текст). Здесь же можно изменить некоторые дополнительные параметры, как показано на рис. 8.19.

После щелчка на кнопке Apply (Применить) в документе появится текст, реагирующий на щелчки мышью. Закрывается диалог щелчком на кнопке Close (Закрыть). Если адрес ссылки был введен корректно, с соблюдением знаков пунктуации и без ошибок, то после щелчка мышью на ссылке будет запущен броузер и откроется веб-страница, на которую указывает ссылка.

Именование ссылок – это хорошая практика, потому что впоследствии вы получите возможность переходить к нужной ссылке в документе с помощью Навигатора. Имя ссылки – это короткое, но достаточно содержательное название объекта, которое должно вводиться в поле Name (Имя) диалога Hyperlink (Гиперссылка) до того, как будет нажата кнопка Apply (Применить).

Кроме того, в гиперссылку можно преобразовать существующий текст, выделив нужный участок текста и вызвав диалог Hyperlink (Гиперссылка). В этом случае

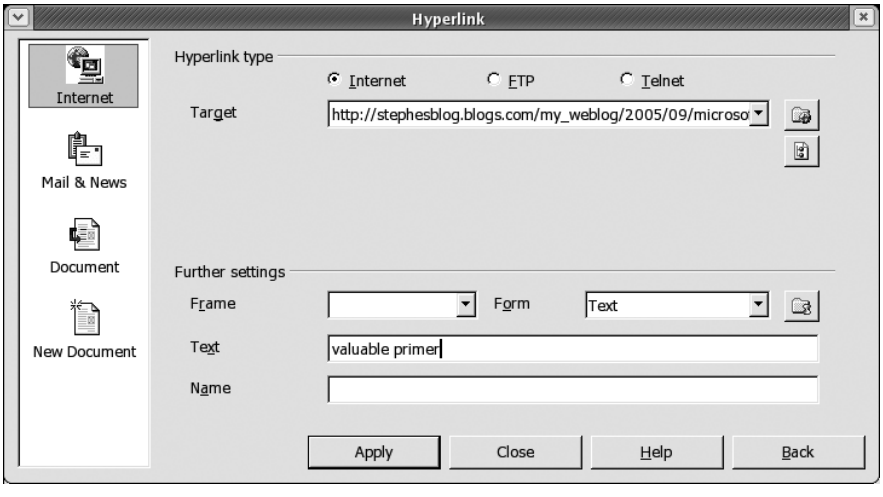


Рис. 8.19. *Вставка гиперссылки*

необходимо заполнить хотя бы поле адреса, после чего можно щелкнуть на кнопке Apply (Применить) и закрыть диалог щелчком на кнопке Close (Закрыть).

Подсчет количества слов

Подсчет количества слов необходимо выполнять таким специалистам, как журналисты, писатели и редакторы, а поскольку слова для них – это хлеб насущный, они могут проявлять беспокойство по поводу отсутствия функции счетчика слов. В OpenOffice, конечно же, имеется счетчик слов, правда, находится он не совсем там, где ожидалось. В MS Word данная функция доступна через пункт меню Tools (Сервис)→Statistics (Статистика), а в OOoWriter доступ к ней осуществляется через пункт меню File (Файл)→Properties (Свойства)→Statistics (Статистика).

Защита документа паролем

Обезопасить документы OOoWriter от доступа посторонних лиц можно, сохраняя файлы с включенной парольной защитой. Для этого достаточно при сохранении файла с помощью пункта меню File (Файл)→Save As (Сохранить как) установить флажок Save with password (Сохранить с паролем) и дважды ввести пароль, который будет запрошен в процессе сохранения документа.

Чтобы отключить защиту паролем, нужно просто выбрать пункт меню File (Файл)→Save As (Сохранить как), снять флажок Save with password (Сохранить с паролем) и выполнить сохранение.

В OOoWriter имеется несколько разновидностей защиты документов от записи исправлений, изменений разделов, рамок, графических элементов, объектов, указателей и таблиц. Дополнительную информацию по этой теме можно найти в подразделе «passwords: protecting content» («пароли: защита содержимого») справочной системы.

Настройка OOoWriter

OpenOffice обладает большим количеством настраиваемых параметров. Короткое знакомство с пятью вкладками в диалоге Tools (Сервис)→Configure (Настройка) (Menu (Меню), Keyboard (Клавиатура), Status Bar (Строка состояния), Toolbars (Панели инструментов), Events (События)) покажет, насколько широкие возможности в настройке предлагает OOoWriter опытным пользователям и системным администраторам.

Настройка панелей инструментов OOoWriter

Особенности технологического процесса и природа бизнеса каждой организации определяют форму представления комплекта инструментальных средств. Таким образом, большие возможности в настройке инструментальных панелей могут помочь системным администраторам и опытным пользователям вывести наверх наиболее часто используемые инструментальные панели или объекты и тем самым увеличить производительность как свою собственную, так и других пользователей в рабочей группе.

В дополнение к инструментальным панелям, имеющимся в OOoWriter по умолчанию (главное меню, панель функций, контекстная панель и главная инструментальная панель), доступны следующие панели: контекстная панель таблицы, контекстная панель нумерации, контекстная панель врезки, панель элементов управления, контекстная панель текста, контекстная панель рисунка, контекстная панель объектов Безье, контекстная панель графических объектов, контекстная панель текста/Веб, контекстная панель кадра/Веб, контекстная панель графических объектов/Веб, контекстная панель объектов/Веб и определяемая пользователем № 1.

Любая из панелей инструментов (за исключением главного меню) может быть скрыта, для этого надо лишь снять флажок напротив названия панели в верхней половине контекстного меню, которое открывается щелчком правой кнопки мыши на пустом пространстве в любой панели инструментов (рис. 8.20).

Помимо этого имеется возможность переупорядочить элементы управления на панелях инструментов и придать им вид, соответствующий личным или коллек-

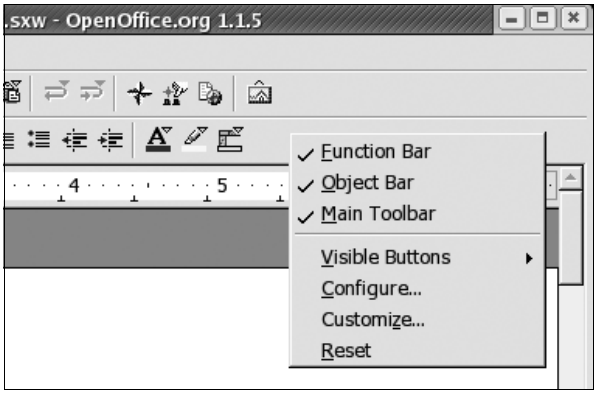


Рис. 8.20. *Контекстное меню панелей инструментов*

тивным предпочтениям, для чего нужно выбрать соответствующий пункт в нижней половине контекстного меню: Visible Buttons (Показать кнопки), Configure (Настройка), Customize (Настройка панелей инструментов) или Reset (Восстановить). Изменения, выполненные с помощью этих команд, оказывают влияние на конкретную панель инструментов – ту, где было вызвано контекстное меню щелчком правой кнопки мыши.

Добавление ярлыка OOoWriter на рабочий стол

Запуск OOoWriter с помощью двойного щелчка на ярлыке выполнить обычно проще, чем отыскивать нужный пункт в системе каскадных меню. Можно добавить на рабочий стол ярлыки всех компонентов, входящих в состав OpenOffice. Ниже приводится сценарий добавления ярлыка OOoWriter на рабочий стол или на инструментальную панель рабочего стола, которая расположена вдоль верхнего или нижнего края экрана. Пример рассчитан на окружение рабочего стола GNOME, для KDE последовательность действий будет иной.

Нужно щелчком правой кнопки мыши на инструментальной панели рабочего стола открыть контекстное меню и выбрать пункт Add to Panel (Добавить на панель)→Launcher from menu (Запуск приложения из меню)→Office (Офис)→OpenOffice Text Document (OpenOffice Writer) (последовательность пунктов меню может отличаться в разных дистрибутивах). В результате на панели появится ярлык запуска OOoWriter. Чтобы добавить ярлыки OOoCalc или OOoImpress, нужно на последнем шаге выбрать пункт OpenOffice Spreadsheet (Электронную таблицу) или OpenOffice Presentation (Презентацию).

Чтобы добавить аналогичный ярлык на рабочий стол, просто перетащите мышью только что созданные ярлыки с панели на рабочий стол. В результате на рабочем столе будет создана копия ярлыка. После этого, если возникнет такое желание, ярлыки на панели можно удалить, для чего нужно щелкнуть правой кнопкой мыши на ярлыке и выбрать пункт контекстного меню Remove From Panel (Убрать с панели). Чтобы удалить ярлык с рабочего стола, нужно щелкнуть правой кнопкой мыши на ярлыке и выбрать пункт контекстного меню Move to Trash (Переместить в корзину).

Изменение непопулярных настроек по умолчанию

В пакете OpenOffice по умолчанию включены функции автодополнения слов, замены некоторых символов и подстановка символа верхнего регистра в начале предложения. Если подобные действия кажутся вам чересчур навязчивыми, положение дел можно легко исправить, отрегулировав параметры этих функций или вообще отключив их.

Автоматическое дополнение слов (отключение). Функция автоматического дополнения слов в OOoWriter по умолчанию включена. Одни пользователи находят неудобным или раздражающим пользоваться текстовым процессором, который вставляет концовки слов до того, как они будут напечатаны. Другие же вполне довольны этой функцией и оставляют настройки по умолчанию.

Если вы находите эту функцию удобной, просто нажимайте клавишу Enter, когда предлагаемая концовка совпадает с тем, что вы собираетесь напечатать, в противном случае нажмите клавишу пробела, чтобы отказаться от предлагаемого варианта.

Чтобы выключить функцию автоматического дополнения слов, нужно выбрать пункт меню Tools (Сервис)→AutoCorrect/AutoFormat (Автозамена/Автоформат)→Word Completion (Дополнение слова), снять флажок Enable word completion (Дополнять слова) и щелкнуть на кнопке OK.

Автозамена (отключение). Если функция автозамены кажется вам слишком агрессивной, например, когда она замещает во время ввода последовательности символов (c) значком авторского права ©, у вас имеется две возможности, чтобы изменить ее поведение: отредактировать список замен или вообще отключить эту функцию.

Редактирование списка замен выполняется очень просто. Нужно выбрать пункт меню Tools (Сервис)→AutoCorrect/AutoFormat (Автозамена/Автоформат) и в открывшемся диалоге перейти на вкладку Replace (Заменить). Далее выделите требуемую строку и щелкните на кнопке Delete (Удалить) или введите иное значение в поле With (Заменить на).

Чтобы вообще отключить функцию автозамены, нужно выбрать пункт меню Tools (Сервис)→AutoCorrect/AutoFormat (Автозамена/Автоформат) и перейти на вкладку Options (Параметры). В самом верху списка находится пункт Use replacement table (Применять таблицу замен) с двумя флажками. Сняв оба флажка в колонках [M] ([И]) и [T] ([В]), можно отключить автозамену в определенных случаях. Здесь же можно отключить другие параметры, управляющие функцией замены, снимая соответствующие флажки в колонках [M] ([И]) и [T] ([В]) у остальных элементов списка.

Настройка функции автозамены. Обратите внимание: в диалоге Tools (Сервис)→AutoCorrect/AutoFormat (Автозамена/Автоформат) крайняя левая вкладка, Replace (Заменить), содержит список замен по умолчанию. Этот список основан на наиболее известных ошибках, допускаемых при нажатии клавиш, и часто используемых символах (таких, как знак авторского права). Функция автозамены может оказать существенную помощь в составлении документов, особенно если настроить список замены, добавив в него свои комбинации слов и символов замены. Чтобы добавить новый элемент списка, нужно заполнить поля Replace (Заменить) и With (Заменить на), а затем щелкнуть на кнопке New (Создать). Чтобы удалить какой-либо элемент списка, сначала нужно его выделить, а затем щелкнуть на кнопке Delete (Удалить).

Автоматическое исправление регистра символов (отключение). Функция автоматического исправления регистра символов в OOoWriter выбирает верхний регистр для некоторых символов, которые вводятся вслед за символом точки. Кроме того, она исправляет две прописные буквы в начале слова, изменяя регистр второй прописной буквы на нижний. В большинстве случаев это очень удобно, поскольку человек иногда не успевает нажать клавишу Shift (подобное случается на удивление часто). Однако, когда вводятся сокращения или акронимы, которые требуют наличия как минимум двух первых заглавных символов, подобные действия функции автозамены становятся нежелательными.

Если функция исправления регистра символов раздражает пользователя или нарушает привычный стиль работы, ее можно отключить, для чего нужно выбрать пункт Tools (Сервис)→AutoCorrect/AutoFormat (Автозамена/Автоформат) и в открывшемся диалоге перейти на вкладку Options (Параметры). Здесь следует убрать

флажки в колонках [M] ([И]) и [T] ([В]) во втором сверху элементе списка, Correct Two Initial Capitals (Исправлять две прописные буквы в начале слова), и в третьем, Capitalize the first letter of every sentence (Начинать каждое предложение с прописной буквы).

Автоматическое исправление регистра символов (добавление исключений). Функция автоматического исправления регистра символов может быть очень удобной, когда ее действия совпадают со стилем работы человека. Поэтому лучше не отключать эту функцию, а лишь добавить некоторые исключения, чтобы заставить ее работать не против человека, а за него. Чтобы настроить исключения для функции автоматического исправления регистра символов, нужно выбрать пункт меню Tools (Сервис)→AutoCorrect/AutoFormat (Автозамена/Автоформат) и в открывшемся диалоге перейти на вкладку Exceptions (Исключения).

На вкладке Exceptions (Исключения) можно добавить свои сокращения, используемые достаточно часто. Для этого нужно ввести требуемое сокращение в поле Abbreviations (no subsequent capitals) (Сообщения, после которых не следует прописных букв) (в верхней части диалога) и щелкнуть на кнопке New (Создать). Эти исключения разрешают функции автоматического исправления изменять регистр первого символа в новом предложении, но запрещают исправлять регистр символа, следующего за одним из перечисленных в списке исключений.

Здесь же, на вкладке Exceptions (Исключения), можно добавить свои исключения к списку слов или акронимов, которые могут начинаться с двух прописных букв. Примерами могут служить элементы, уже присутствующие в списке. Наряду с собственными исключениями в этот список можно добавить такие обозначения, как «OOo», «OOoWriter» и множество вариаций на эту тему.

OpenOffice Calc

Программа OpenOffice Calc (известная также под названием OOoCalc) – это приложение электронных таблиц, входящее в состав пакета OpenOffice. Пользователи, знакомые с последними версиями Microsoft Excel, будут чувствовать себя в OOoCalc как дома.

Управление файлами

Операции открытия, сохранения, отправки и экспорта файлов в OOoCalc ничем не отличаются от аналогичных операций в OOoWriter, описанных выше во всех подробностях.

Ввод меток (текста)

Вводить в ячейки текст, а не число следует точно так же, как и в MS Excel: последовательность символов должна начинаться с символа одинарной кавычки (') , а по окончании ввода должна нажиматься клавиша Enter.

Автозаполнение

В OOoCalc существует возможность быстро заполнить числами строку или столбец всего одним движением мыши. После ввода некоторого числа, например 1 в ячейку A1, нужно выделить ячейку с числом, щелкнув на ней, затем поместить указатель мыши на маленький черный квадратик в правом нижнем углу

ячейки, нажать левую кнопку мыши и, удерживая ее, переместить указатель мыши вниз или вправо. После того как кнопка будет отпущена, выделенная область будет заполнена последовательностью чисел.

Ввод простых формул

Ввод формул – это самое основное, что должен знать опытный пользователь электронной таблицы. Формулы всегда начинаются со знака равенства (=). Например, чтобы вычислить результат 1 + 1, нужно ввести последовательность = 1 + 1 и нажать клавишу Enter.

Чтобы вычислить результат на основе содержимого других ячеек, нужно ввести символ = в ячейку, в которой будет находиться результат, а затем щелкнуть на первой ячейке в формуле. В результате вокруг ячейки появится красная рамка. Далее нужно ввести оператор, например +, и щелкнуть на второй ячейке. Вокруг этой ячейки также появится красная рамка. Можно продолжать и дальше вводить операторы, сопровождая их выбором ячеек или вводом других значений. По завершении ввода формулы следует нажать клавишу Enter, и результат появится в ячейке.

Обратите внимание: поле ввода формулы, расположенное выше области ячеек электронной таблицы, содержит только что созданную формулу. Ту же самую формулу можно создать другим способом, для этого достаточно просто ввести ее прямо в поле ввода формулы. Сначала нужно выбрать требуемую ячейку результата, щелкнув на ней один раз, а затем щелкнуть на поле ввода формул, ввести формулу с клавиатуры и нажать клавишу Enter.

Суммирование чисел в столбце

Чтобы быстро просуммировать существующий столбец чисел, нужно щелчком мыши выделить пустую ячейку, куда будет записываться результат, и щелкнуть на знаке «сигма» (?) в панели формул. В результате вокруг ближайшего столбца, который расценивается как наиболее вероятный кандидат на суммирование, появится синяя рамка. Если группа чисел была выделена правильно, останется только нажать клавишу Enter, и сумма появится в ячейке результата. Если группа чисел оказалась выделена неправильно, тогда можно захватить левой кнопкой мыши маленький синий квадрат в правой нижней части подсвеченного столбца и откорректировать границы группы. После этого можно нажать клавишу Enter.

Перемещение содержимого ячеек

В OOoCalc проще переместить группу ячеек, чем одну ячейку. Эта задача одна из тех, что доставляют пользователям массу неприятностей, когда они только привыкают к новой среде OOoCalc, но стоит только выполнить ее несколько раз, как она становится очень простой.

Чтобы переместить группу ячеек, нужно просто выделить ту, что расположена в одном из четырех углов диапазона, и, удерживая левую кнопку, переместить указатель к противоположному углу диапазона по диагонали. Когда таким образом будет выделена нужная область, можно будет отпустить левую кнопку мыши. После этого надо поместить указатель мыши в пределы выделенной области, нажать левую кнопку мыши и, удерживая ее, переместить область ячеек

в новое место. После того как вы отпустите кнопку мыши, группа ячеек останется на новом месте.

Процедура перемещения единственной ячейки выполняется точно так же, вот только выделить единственную ячейку для начинающих пользователей оказывается непосильной задачей. Чтобы выделить единственную ячейку, следует установить на нее указатель мыши, нажать левую кнопку и, удерживая ее, начать выделять область ячеек, затем, не отпуская кнопку мыши, вернуть указатель мыши в первоначальное положение и отпустить кнопку. После этого можно будет ухватить выделенную ячейку левой кнопкой мыши и перетащить в нужное место.

В MS Office операция перемещения ячейки выполняется в один этап, тогда как в OOoCalc требуется выполнить два этапа: сначала выделить ячейку и лишь потом переместить ее. Процедура в OOoCalc более сложная, но в конечном итоге более эффективная, и ее совсем не сложно освоить и запомнить (потому что старый метод быстро забудется).

Изменение ширины столбцов и высоты строк

Чтобы изменить ширину столбца, нужно подвести указатель мыши к боковой границе столбца в области заголовка, то есть там, где находятся метки столбцов A, B, C и т. д. Обратите внимание: указатель мыши будет превращаться в двунаправленную горизонтальную стрелку при перемещении через границы столбцов. Пока указатель мыши имеет вид двунаправленной стрелки, просто нажмите в этот момент левую кнопку мыши и, не отпуская ее, переместите границу столбца влево или вправо, чтобы увеличить или уменьшить его ширину. Чтобы вернуть ширину столбца в значение по умолчанию, нужно щелкнуть правой кнопкой мыши на его заголовке, из появившегося контекстного меню вызвать диалог Column Width (Ширина столбцов), установить в нем флажок Default value (По умолчанию) и щелкнуть на кнопке OK. После этого вернется ширина столбца по умолчанию (0,89 дюйма – 2,27 сантиметра).

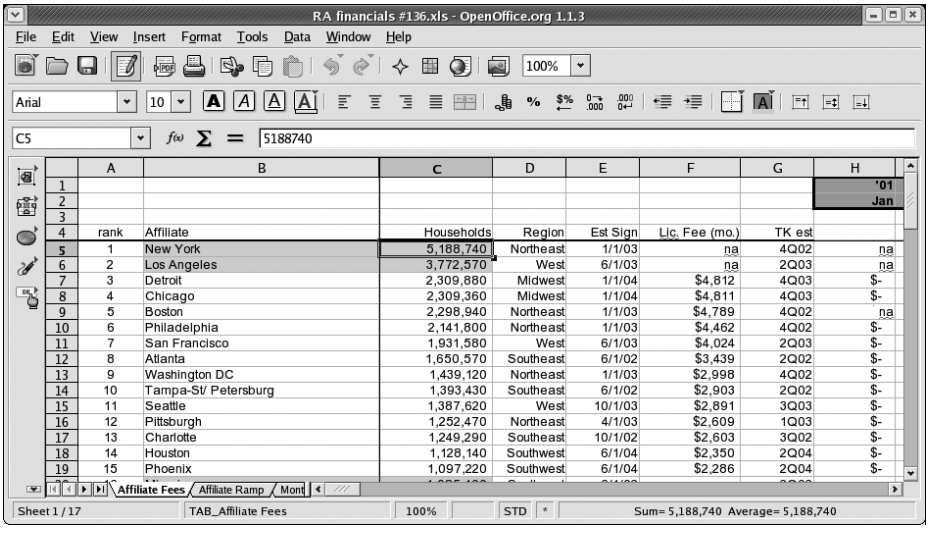
Изменение высоты строк производится аналогичным образом, только указатель мыши надо подводить к верхней или нижней границе строки в области заголовков строк, с левой стороны страницы. Процедура возврата высоты строки в значение по умолчанию также аналогична процедуре восстановления ширины столбца, только вызывать контекстное меню необходимо на заголовке строки, с левой стороны страницы.

Объединение ячеек

Чтобы объединить несколько ячеек, необходимо сначала их выделить, а затем выбрать пункт главного меню Format (Формат)→Merge Cells (Объединение ячеек)→Define (Объединить). В результате появится единственная ячейка, вмещающая содержимое всех выделенных ячеек. Приложение OOoCalc способно распознавать содержимое ячеек и правильно объединять их, например, если один столбец содержит слово «Июнь», а другой содержит число 3, при их объединении получится дата 03.06, сопровождаемая текущим годом.

Фиксация и разбиение окна

По крупным таблицам бывает очень сложно перемещаться, потому что строки и столбцы скрываются из поля зрения. Команды меню Window (Окно)→Freeze



	A	B	C	D	E	F	G	H
1								'01
2								Jan
3								
4	rank	Affiliate	Households	Region	Est Sign	Lic. Fee (mo.)	TK est	
5	1	New York	5,188,740	Northeast	1/1/03	na	4Q02	na
6	2	Los Angeles	3,772,570	West	6/1/03	na	2Q03	na
7	3	Detroit	2,309,880	Midwest	1/1/04	$4,812	4Q03	$-
8	4	Chicago	2,309,360	Midwest	1/1/04	$4,811	4Q03	$-
9	5	Boston	2,298,940	Northeast	1/1/03	$4,789	4Q02	na
10	6	Philadelphia	2,141,800	Northeast	1/1/03	$4,462	4Q02	$-
11	7	San Francisco	1,931,580	West	6/1/03	$4,024	2Q03	$-
12	8	Atlanta	1,650,570	Southeast	6/1/02	$3,439	2Q02	$-
13	9	Washington DC	1,439,120	Northeast	1/1/03	$2,998	4Q02	$-
14	10	Tampa-St/ Petersburg	1,393,430	Southeast	6/1/02	$2,903	2Q02	$-
15	11	Seattle	1,387,620	West	10/1/03	$2,891	3Q03	$-
16	12	Pittsburgh	1,252,470	Northeast	4/1/03	$2,609	1Q03	$-
17	13	Charlotte	1,249,290	Southeast	10/1/02	$2,603	3Q02	$-
18	14	Houston	1,128,140	Southwest	6/1/04	$2,350	2Q04	$-
19	15	Phoenix	1,097,220	Southwest	6/1/04	$2,286	2Q04	$-

Рис. 8.21. *Перед фиксированием заголовков столбцов и строк*

(Фиксировать) и Window (Окно)→Split (Разбить) позволяют зафиксировать положение заголовков строк и столбцов на месте при прокручивании остальной части электронной таблицы.

Чтобы зафиксировать заголовки столбцов и строк, нужно щелкнуть на ячейке, от которой должен проявляться эффект фиксации, и выбрать пункт меню Window (Окно)→Freeze (Фиксировать). В результате в этом пункте меню появится галочка, а области выше и левее выбранной ячейки окажутся зафиксированы. Зафиксированные области отделяются от остальной части таблицы простыми линиями, как показано на рис. 8.21.

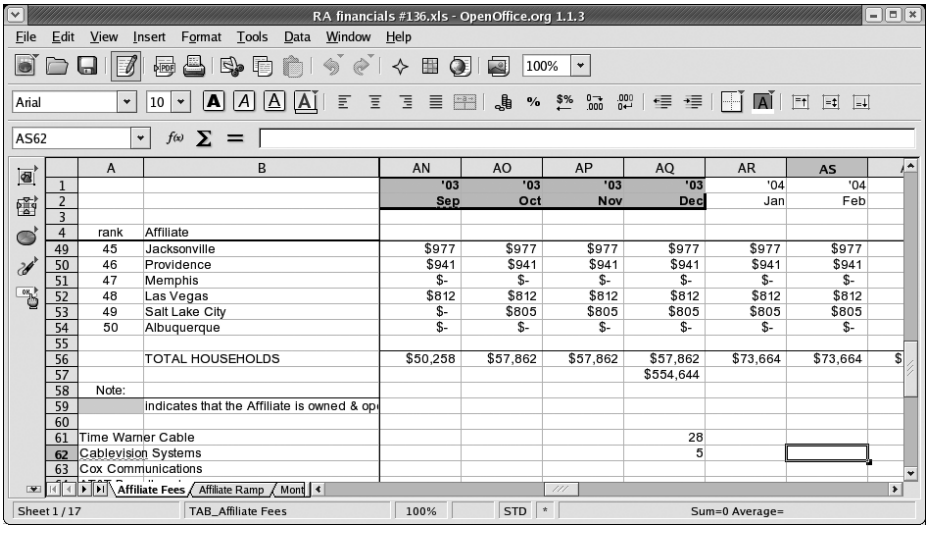
Теперь можно попробовать перемещаться по таблице вниз и вправо. Обратите внимание: на рис. 8.22 показано, как заголовки столбцов остаются на месте в поле зрения при перемещении вниз по таблице. Аналогичный эффект наблюдается с заголовками строк при прокручивании таблицы вправо.

Еще один интересный способ оставить в поле зрения часть таблицы: вместо пункта меню Window (Окно)→Freeze (Фиксировать) выбрать пункт Window (Окно)→Split (Разбить). После этого при прокручивании одна из областей будет перемещаться вместе с одной из соседних областей, в зависимости от того, в каком направлении она прокручивается.

Чтобы отменить фиксацию или разбиение, достаточно еще раз выбрать соответствующий пункт главного меню. Параметры, отвечающие за фиксирование и разбиение, сохраняются в документе и могут «путешествовать» вместе с ним.

Предварительный просмотр деления на страницы

Режим предварительного просмотра деления на страницы позволяет увидеть, как разместится документ на страницах при печати. Чтобы включить режим предварительного просмотра деления на страницы, нужно выбрать пункт Page



	A	B	AN	AO	AP	AQ	AR	AS
1			'03	'03	'03	'03	'04	'04
2			Sep	Oct	Nov	Dec	Jan	Feb
3								
4	rank	Affiliate						
49	45	Jacksonville	$977	$977	$977	$977	$977	$977
50	46	Providence	$941	$941	$941	$941	$941	$941
51	47	Memphis	$-	$-	$-	$-	$-	$-
52	48	Las Vegas	$812	$812	$812	$812	$812	$812
53	49	Salt Lake City	$-	$805	$805	$805	$805	$805
54	50	Albuquerque	$-	$-	$-	$-	$-	$-
55								
56		TOTAL HOUSEHOLDS	$50,258	$57,862	$57,862	$57,862	$73,664	$73,664
57						$554,644		
58	Note:							
59		Indicates that the Affiliate is owned & op						
60								
61	Time Warner Cable					28		
62	Cablevision Systems					5		
63	Cox Communications							

Рис. 8.22. *Заголовки строк и столбцов после фиксации*

Break View (Предварительный просмотр деления на страницы) в меню View (Вид). В результате в этом пункте меню появится галочка. Чтобы выключить режим предварительного просмотра деления на страницы, нужно повторно выбрать этот же пункт меню.

Можно быстро изменять порядок деления на страницы, перетаскивая внешние синие линии, чтобы покрыть требуемую область таблицы, а также перемещая линии, разделяющие отдельные страницы, чтобы добиться вывода определенных столбцов и строк на одной странице. Кроме того, режим предварительного просмотра деления на страницы предоставляет возможность просматривать большие таблицы и перемещаться по ним с более отдаленной перспективы.

Настройка области печати

Новая таблица изначально не имеет установленной области печати. Такая таблица отображается в виде серой области в режиме предварительного просмотра деления на страницы. Чтобы установить область печати крупноформатной таблицы, сначала необходимо установить режим предварительного просмотра деления на страницы. Затем нужно выделить область, которая должна выводиться на печать, щелкнув на ячейке в одном из углов будущей области печати и растянув мышью область выделения до нужных размеров. И наконец, щелкнуть правой кнопкой мыши на выделенной области и выбрать пункт контекстного меню Set a Print Range (Определить диапазон печати). Как вариант выделенную область можно преобразовать в область печати, выбрав пункт главного меню Format (Формат)→Print Ranges (Область печати)→Define (Определить). Любое содержимое таблицы, не попадающее в установленную область, выводиться на принтер не будет.

Если область печати уже была определена и необходимо изменить ее размеры, можно просто захватить мышью угол синей рамки (или одну из сторон) и перетащить его так, чтобы в область рамки попали все ячейки новой области печати.

Чтобы захватить синюю рамку мышью, сначала попробуйте переместить указатель мыши над этой рамкой – вы увидите, как указатель мыши превратится в двунаправленную стрелку. Затем нажмите левую кнопку мыши и, удерживая ее, перетащите сторону или угол рамки в новое место.

Функции

В OOoCalc имеется большое количество разнообразных функций, включая функции финансовых расчетов, работы с базами данных, со временем (дата и время), с массивами, выполнения статистических расчетов, информационные, логические, математические и текстовые.

Функции OOoCalc, их синтаксис и форматы аргументов описаны в справочной системе, доступной через меню Help (Справка)→Contents (Содержание). После выбора этого пункта меню откроется окно справочной системы. Затем на вкладке Index (Индекс) нужно ввести имя функции в поле Search Term (Искомое понятие) и щелкнуть на кнопке Enter (Показать) (можно просто дважды щелкнуть на названии функции в левой панели), в результате будет выведена справочная информация об этой функции. Пример вывода справочной информации о финансовой функции PV, которая вычисляет текущую стоимость инвестиций после ряда платежей, приводится на рис. 8.23. Функция PV – это функция электронной таблицы, которая пользуется определенной популярностью у специалистов, работающих в области кредитования.

При вводе имени функции в ячейку никогда не следует забывать вставлять начальный символ знака равенства (=). На рис. 8.24 приводится пример того, как должна выглядеть формула на основе функции PV, с корректным набором входных аргументов: `=PV(B1; B2; B3)`.

Функция на рис. 8.24 – это типичное решение задачи вычисления стоимости кредита. Например, если ваш банк предоставляет вам ссуду сроком на 30 лет под 5% годовых и известно, что ежемесячно вы будете выплачивать ровно $1500, появляется вопрос: «Какова будет стоимость ссуды при ежемесячных выплатах $1500?».

Функция PV как раз и решает поставленную задачу. С точки зрения плательщика, платежи по ссуде, выплачиваемые банку, являются исходящими, и потому они представляются отрицательной величиной. По этой причине сумма ежемесячных выплат должна указываться со знаком «минус», в противном случае результат получится отрицательным. Количество периодов составляет 30 лет по 12 месяцев в каждом, всего 360 периодов, а процентная ставка за период при 5% годовых составит 0,42% в месяц, как указано в примере на рис. 8.24.

Можно также было бы воспользоваться функцией PMT (Payment – платежи), чтобы определить, какую сумму придется выплачивать ежемесячно в погашение кредита за дом своей мечты стоимостью 10 млн. евро.

Можно было бы также вместо имен ячеек передать функции абсолютные значения. В этом случае формула примет такой вид:

```
=PV(.0042;360;-1500)
```

Однако использование ссылок на ячейки оставляет свободу действий для поиска альтернативных вариантов или для анализа результатов, получающихся при изменении одной из переменных в определенных границах.

Рис. 8.23. *Описание функции PV (Present Value – стоимость инвестиций) в справочной системе*

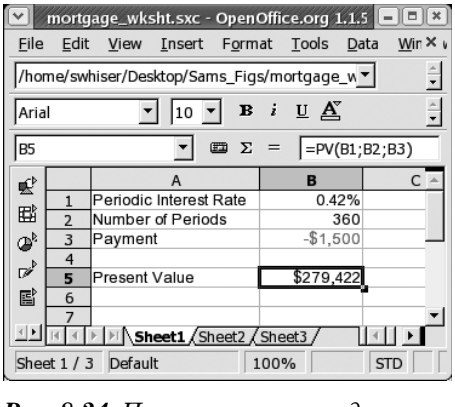


Рис. 8.24. *Пример решения задачи вычисления стоимости инвестиций*

Рабочие листы, или просто листы

Файл электронной таблицы OOoCalc (иногда называемый *книгой*) по умолчанию содержит три листа, но их количество может быть доведено до 256.

На рис. 8.25 показаны три стандартных листа, присутствующие в файле электронной таблицы по умолчанию. Обратите внимание: на рисунке по окраске закладки, соответствующей первому листу, видно, что лист с номером 1 является текущим. Серая окраска вкладок листов с номерами 2 и 3 указывает, что они присутствуют в книге, но в настоящий момент скрыты.

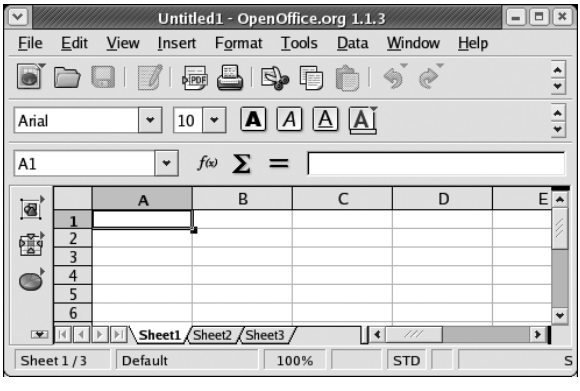


Рис. 8.25. *Три листа книги*

Чтобы переместиться на другой лист, нужно щелкнуть на вкладке листа, после чего он станет текущим.

Чтобы добавить новый лист, нужно щелкнуть правой кнопкой мыши на области с вкладками и из контекстного меню выбрать пункт Insert Sheet (Добавить лист), в результате чего на экране появится диалог Insert Sheet (Вставить лист). В диалоге нужно определить название, положение и количество новых листов. Обратите внимание: здесь имеется возможность добавить сразу несколько листов. Можно также вставить листы из другого файла; после того как требуемый файл будет найден, названия его листов появятся в списке, из которого можно будет выбрать нужные.

Чтобы удалить лист из книги, сначала нужно сделать этот лист текущим, щелкнув на его вкладке. Затем нужно щелкнуть правой кнопкой мыши на вкладке листа и выбрать в контекстном меню пункт Delete Sheet (Удалить). В появившемся диалоге запроса на подтверждение операции удаления надо будет ответить Yes (Да).

Чтобы переименовать текущий лист, необходимо щелкнуть правой кнопкой мыши на вкладке листа и выбрать в контекстном меню пункт Rename Sheet (Переименовать). В результате этого действия будет активизирован диалог Rename Sheet (Переименовать лист), где в поле Name (Имя) можно будет ввести новое название листа.

Чтобы выбрать одновременно несколько листов, нужно нажать клавишу Ctrl и, удерживая ее, щелкнуть на вкладке каждого отбираемого листа.

Выбор сразу нескольких листов может потребоваться, например, при вводе названий заголовков столбцов, которые должны присутствовать на нескольких

листах. Это экономит время, необходимое на ввод одной и той же информации в нескольких листах.

Если книга состоит из большого числа листов и необходимо выбрать некоторый последовательный диапазон листов, тогда нужно сделать текущим крайний левый лист диапазона. Затем, удерживая нажатой клавишу Shift, нужно щелкнуть на вкладке крайнего правого листа в диапазоне. В результате этих действий окажутся выбранными все листы из требуемого диапазона.

Чтобы снять выделение с группы листов, нужно при нажатой клавише Shift щелкнуть на первом листе диапазона (в данном случае крайний левый лист).

Чтобы снять выделение с выбранного листа (не являющегося текущим, который всегда считается выбранным), нужно при нажатой клавише Ctrl щелкнуть на вкладке этого листа.

Если на экране не умещаются вкладки всех листов, тогда, чтобы сделать видимой вкладку нужного листа, необходимо воспользоваться кнопками прокрутки, расположенными слева от панели вкладок.

Сортировка данных

Чтобы отсортировать список или диаграмму с числовой или текстовой информацией, сначала нужно выделить полный диапазон сортируемых ячеек (за исключением данных, таких как сумма, которые не должны участвовать в сортировке) и затем выбрать пункт главного меню Data (Данные)→Sort (Сортировка). В результате на экране появится диалог Sort (Сортировка), где можно будет определить порядок сортировки и ряд других параметров.

В примере на рис. 8.26 мы решили переупорядочить данные об ответах респондентов таким образом, чтобы наиболее часто встречающиеся ответы располага-

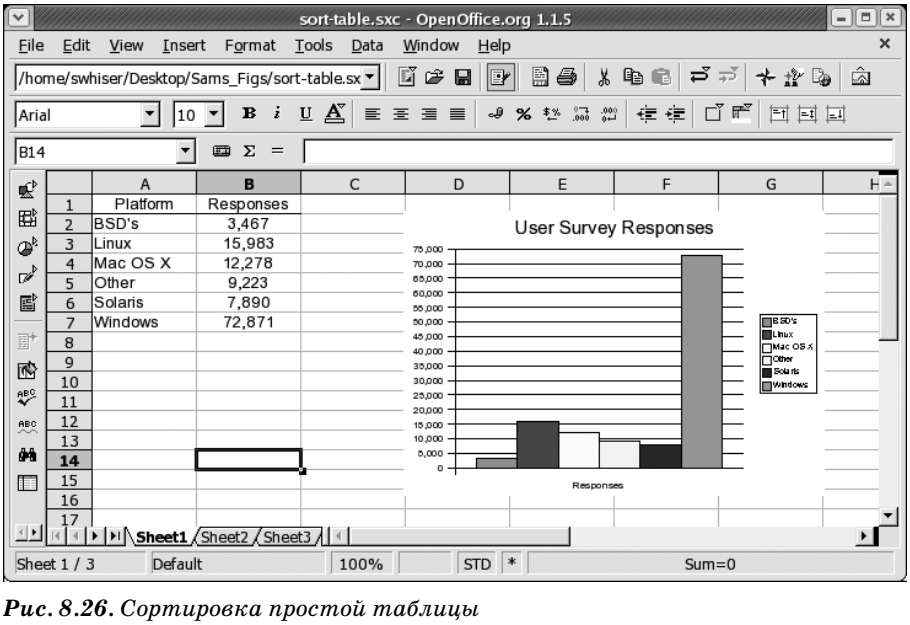


Рис. 8.26. *Сортировка простой таблицы*

лись в начале списка. Поэтому в диалоге Sort (Сортировка) была выбрана сортировка по столбцу Responses per Platform (содержащему числа) в порядке убывания (переключатель Descending (По убыванию) справа). Затем была нажата кнопка OK. Обратите внимание: диаграмма автоматически была перестроена в соответствии с новым порядком сортировки (рис. 8.27).

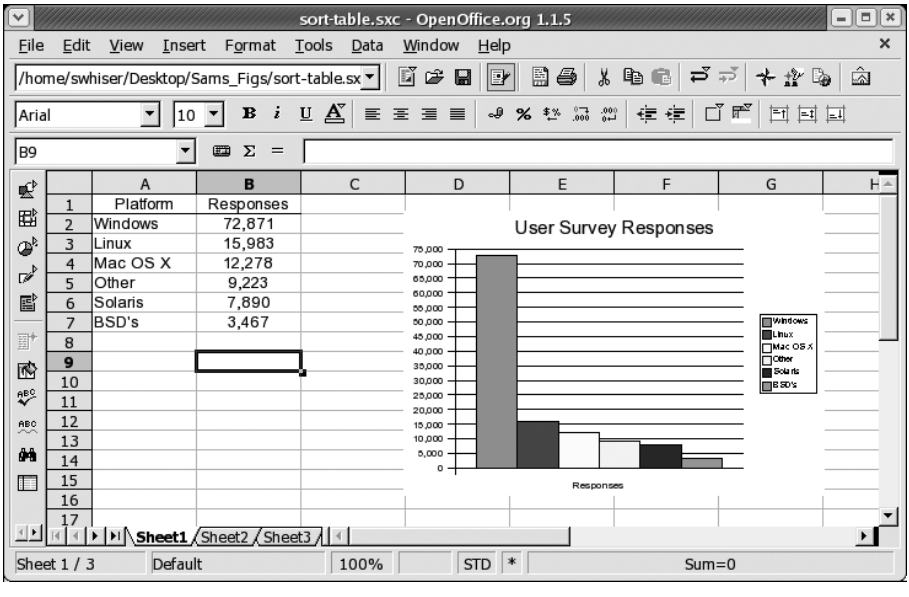


Рис. 8.27. *Таблица (и диаграмма) после сортировки*

Источники данных

Вместо того чтобы обзаводиться базой данных собственного формата, приложение OOoCalc было разработано так, чтобы обеспечить возможность взаимодействовать с различными типами внешних баз данных. Источники данных (Data Sources) – это название набора функциональных возможностей, существующих в OOoCalc и предназначенных для организации взаимодействия с базами данных и связывания электронных таблиц, форм и отчетов с информацией, содержащейся в базах данных. OOoCalc предоставляет возможность организовать получение информации, например, из баз данных MySQL или Adabas D, а также из других источников данных, включая MS Outlook, Outlook Express, Mozilla и другие.

Чтобы в OOoCalc вызвать окно обзора источников данных, нужно выбрать пункт главного меню Tools (Сервис)→Data Sources (Источники данных) или просто нажать клавишу F4. Чтобы закрыть окно обзора, нужно еще раз нажать клавишу F4.

После такого многообещающего введения нам крайне неудобно заявлять, что обсуждение темы источников данных выходит за рамки этого раздела. Это очень неприятно, потому что взаимодействие с базами данных стало обыденным делом для пользователей офисных приложений, имеющих доступ в Web. Кроме того, взаимодействие с базами данных в OpenOffice – одно из самых быстроразвивающихся направлений, и это позволяет рассчитывать, что каждая новая версия

OpenOffice,будет предлагать все более простые и легкие для рядового пользователя средства доступа к базам данных.

Макросы

Обсуждение вопросов создания и управления макросами в OOoCalc выходит за рамки этой книги. Однако мы можем предложить вам информацию общего характера, которая будет полезной для желающих пользоваться макросами. Макроопределения можно использовать в любом из компонентов OpenOffice (и в MS Office), но здесь мы будем иметь дело только с макросами в OOoCalc (и MS Excel).

Для написания макроопределений в OOoCalc используется свой собственный язык, который называется OpenOffice Basic (или StarBasic). Этот язык отличается от языка, используемого корпорацией Microsoft в MS Office, который называется Visual Basic (или VBA).

Макроопределения на языке VBA не могут исполняться в OOoCalc, что создает серьезный барьер для миграции с MS Excel на OOoCalc для тех пользователей, кто обладает электронными таблицами MS Excel с большим числом макроопределений на языке VBA. В настоящее время, чтобы перенести таблицу с макросами на языке VBA из MS Excel в OOoCalc, эти макросы придется переписать на языке StarBasic.

Компания Sun Microsystems обещала выпустить средство преобразования макросов с языка VBA на язык StarBasic, чтобы облегчить автоматическое преобразование макроопределений.

В настоящее время OOoCalc по умолчанию оставляет макроопределения VBA в неприкосновенности, чтобы они оставались доступными при сохранении электронных таблиц в формате MS Excel. Есть три возможности:

- повторно импортировать электронную таблицу в Excel, чтобы запустить сохраненные макроопределения VBA;
- сохранить макроопределения VBA, чтобы позднее вручную переписать их на языке StarBasic;
- оставить макроопределения VBA неиспользованными в OOoCalc, чтобы позднее, когда Sun Microsystems наконец выпустит свой конвертор, преобразовать их в макросы на языке StarBasic.

Поскольку макроопределения на языке VBA не могут работать в OOoCalc, вирусы, связанные с ними, не представляют никакой угрозы при работе с OOoCalc. Если по каким-либо причинам (например, из соображений безопасности или просто потому, что они стали не нужны) желательно удалить макросы VBA при импорте файлов Excel, достаточно выключить параметры по умолчанию в диалоге, открывающемся при выборе пункта меню Tools (Сервис)→Options (Параметры)→Load/Save (Загрузка/сохранение)→VBA Properties (Свойства VBA).

Если макроопределения представляют для вас определенный интерес, за дополнительной информацией по этой теме можно обратиться к документу «OpenOffice Basic programmer's Guide» на сайте *http://docs.sun.com/db/doc/817-1826?q=star+basic*.

OpenOffice Impress

OpenOffice Impress (известное также под названием OOoImpress) – это приложение презентаций, входящее в состав пакета OpenOffice. Те, кто знаком с последними версиями Microsoft PowerPoint, не будут чувствовать себя потерянными при работе с OOoImpress.

Создание новых презентаций с помощью мастера

Когда запускается компонент OOoImpress – с помощью ярлыка, расположенного на рабочем столе или на панели, из другого модуля OOo или с помощью пункта меню File (Файл)→New (Создать)→Presentation (Презентацию), – на экране появляется мастер презентаций, который поможет в создании презентации «с нуля». С помощью мастера можно открыть уже существующую презентацию или шаблон презентации.

Менее опытные пользователи смогут с помощью мастера пройти весь путь создания новой презентации, опытные же пользователи могут просто щелкнуть на кнопке Create (Готово) и начать работать с новой пустой презентацией.

Открытие существующей презентации

Чтобы открыть презентацию, созданную ранее или полученную со стороны, нужно просто щелкнуть на ярлыке файла в окне папки. Ваша система Linux наверняка будет настроена таким образом, чтобы автоматически открывать файлы MS PowerPoint (с расширением *.ppt*) с помощью OOoImpress. По умолчанию файлы презентаций сохраняются в том же формате, в каком они существовали до их открытия (PowerPoint, OOoImpress и т. д.).

Как вариант, можно выбрать пункт главного меню File (Файл)→Open (Открыть) и отыскать в файловой системе существующий файл презентации, чтобы открыть его.

Сохранение презентации

Чтобы сохранить текущую презентацию в ее текущем местоположении и формате, нужно щелкнуть на кнопке Save (Сохранить) (с изображением дискеты) на панели функций, и файл будет сохранен на старом месте в файловой системе. То же самое произойдет, если выбрать пункт главного меню File (Файл)→Save (Сохранить).

Если сохранение файла происходит впервые, будет открыт диалог Save (Сохранить), который позволит выбрать каталог для сохранения и имя файла. Когда выбор сделан, нужно щелкнуть на кнопке Save (Сохранить). По умолчанию диалог Save (Сохранить) предлагает сохранить файл в каталоге *Documents*. Таким образом, пользователю *swhiser* будет предложено сохранить документ в каталоге */home/swhiser/Documents*. Это же правило по умолчанию действует и в других компонентах пакета OpenOffice.

Если возникнет необходимость сохранить файл под другим именем, в другом каталоге или в другом формате, нужно выбрать пункт главного меню File (Файл)→Save As (Сохранить как) и заполнить соответствующим образом поля диалога Save As (Сохранить как).

Форматы экспорта

Одно из основных достоинств OOoImpress – это большое разнообразие форматов файлов, в которые можно экспортировать презентации. В табл. 8.4 приводится перечень различных форматов, в которые возможен экспорт презентаций.

Таблица 8.4. Форматы файлов, поддерживаемые OOoImpres для экспорта

Формат	Название	Расширение файла
BMP	Windows Bitmap	*.bmp*
EMF	Enhanced Metafile	*.emf*
EPS	Encapsulated Postscript	*.eps*
GIF	Graphics Interchange Format	*.gif*
HTML	Hypertext Markup Language	*.html, .htm*
JPEG	Joint Photographic Experts Group	*.jpg, .jpeg, .jfif, .jif, .jpe*
MET	OS/2 Metafile	*.met*
PBM	Portable Bitmap	*.pbm*
PCT	Mac Pict	*.pct*
PDF	Printable Document Format	*.pdf*
PGM	Portable Greymap	*.pgm*
PNG	Portable Network Graphic	*.png*
PPM	Portable Pixel Map	*.ppm*
PWP	Placeware	*.pwp*
RAS	Sun Raster Image	*.ras*
SVG	Scalable Vector Graphics	*.svg*
SVM	StarView Metafile	*.svm*
SWF	Macromedia Flash	*.swf*
SXI	Формат файлов OOoImpress	*.sxi*
TIFF	Tagged Image File Format	*.tif, .tiff*
WMF	Windows Metafile	*.wmf*
XPM	X PixMap	*.xpm*

Экспорт в формат HTML. Среди самых удобных возможностей стоит упомянуть экспорт презентации в формате веб-страниц, или HTML. Данная особенность позволяет безболезненно преобразовать любую презентацию в формат, наилучшим образом подходящий для публикации в Web, чтобы любой желающий мог ознакомиться с материалами презентации из любой точки земного шара.

Для начала нужно выбрать пункт главного меню File (Файл)→Export (Экспорт). В результате будет запущен диалог Export (Экспорт), где можно будет выбрать формат сохраняемого файла «Документ HTML (OpenOffice Impress) (.html; .htm)» в раскрывающемся списке File format (Тип файла), указать имя файла

и определить каталог, куда должны быть сохранены файлы HTML. После этого нужно щелкнуть на кнопке Export (Сохранить), чтобы запустить серию диалогов HTML Export (Экспорт HTML).

Сначала придется выбрать дизайн. В этом месте можно оставить выбор по умолчанию и щелкнуть на кнопке Next (Далее), что вполне подходит для большинства ситуаций. После этого будут предложены различные типы публикаций, которые определяют способ отображения и управления презентациями в Web. Среди типов публикаций имеются: standard HTML format (Стандартный формат HTML), standard HTML with frames (Стандартный HTML с фреймами), automatic (Автоматически) и WebCast (интернет-трансляция) (требует наличия сервера).

В большинстве случаев вполне подойдет выбор по умолчанию, поэтому здесь можно просто щелкнуть на кнопке Next (Далее). На следующей странице диалога будет предложено определить формат графических файлов и разрешение. Здесь же можно включить или выключить экспорт звуковых эффектов. Значения этих параметров также можно оставить по умолчанию. После щелчка на кнопке Next (Далее) будет предложено ввести информацию для титульной страницы нового представления. Здесь следует ввести необходимые сведения о себе, после чего можно щелкнуть на кнопке Next (Далее). На следующей странице будет предложено выбрать стиль элементов навигации, таких как кнопки Forward (Вперед) и Backward (Назад). Если здесь оставить установленным флажок Text only (Только текст) (значение по умолчанию), в качестве элементов навигации будут использоваться текстовые гиперссылки, в противном случае можно будет выбрать один из четырех стилей красочных кнопок.

После щелчка на кнопке Next (Далее) будет открыта последняя страница в процедуре экспорта, где можно выбрать цветовую схему отображения текста. Для первого раза можно оставить значения по умолчанию. В заключение нужно щелкнуть на кнопке Create (Готово), и презентация будет готова для публикации в Web.

Экспорт в формат Macromedia Flash. Не менее важным среди множества форматов является формат Macromedia Flash. Это еще один универсальный формат (наряду с PDF и HTML), который гарантирует, что вашу презентацию можно будет просмотреть с помощью любого веб-броузера (то есть на любом настольном компьютере). Многие из преимуществ, которые дает преобразование презентаций в формат HTML (описанные выше), сохраняются и для формата Flash.

Для экспорта презентации в формат Flash нужно выбрать пункт главного меню File (Файл)→Export (Экспорт), чтобы открыть диалоговое окно Export (Экспорт), где в раскрывающемся списке File format (Тип файла) можно выбрать формат Macromedia Flash (SWF) (.swf)». Если в диалоге Export (Экспорт) каталог или путь к файлу не будут изменены, новая Flash-версия презентации будет сохранена в тот же каталог, где находится оригинальный файл презентации *.sxi*. Далее нужно щелкнуть на кнопке Export (Сохранить), и Flash-версия презентации будет создана.

Представление рабочего пространства в OOoImpress

В OOoImpres существует несколько вариантов представления рабочего пространства. Чтобы изменить вид рабочего пространства, в главном меню следует выбрать пункт View (Вид)→Workspace (Рабочий режим) и далее в дополнительном раскрывающемся меню желаемый вариант представления. Всего имеется пять разных представлений рабочей области: Drawing View (Режим рисования), Outline

View (Режим структуры), Slides View (Режим слайдов), Notes View (Режим примечаний) и Handout View (Режим тезисов). Чаще других используется режим рисования, в котором производятся основные работы по созданию и редактированию презентаций.

Переключение способов представления можно производить гораздо быстрее с помощью маленьких кнопок, выстроенных вдоль правого края окна сверху, как показано на рис. 8.28.

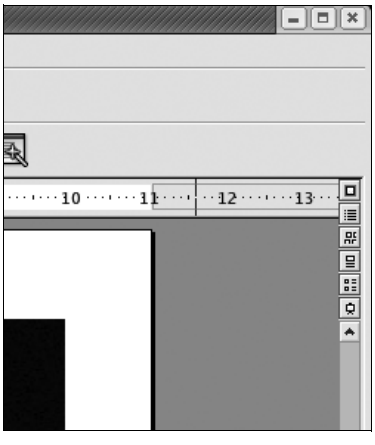


Рис. 8.28. *Кнопки переключения вида рабочего пространства*

Режимы OOoImpress

Режимы – это такие состояния, в которых могут быть выполнены только определенные функции редактирования или установлен определенный способ представления и ориентации.

Всего имеется три режима. Получить к ним доступ и выбрать текущий режим можно через главное меню – это пункты View (Вид)→Slide (Слайд), View (Вид)→ Master (Фон) и View (Вид)→Layers (Слой). Пункт меню, соответствующий активному режиму, отмечается галочкой.

Пользователей OpenOffice постоянно приводит в замешательство тот факт, что выяснение и изменение режима производится из главного меню View (Вид). Хуже того, OOoImpres изменяет параметры представления содержимого меню View (Вид) в зависимости от выбранного режима. Быстро изменить режим можно с помощью трех кнопок, расположенных вдоль нижнего края рабочей области с левой стороны (рис. 8.29).

Из-за сложности изменения способов представления и режимов из главного меню рекомендуется пользоваться кнопками: для изменения режимов предназначены три кнопки вдоль нижней границы рабочей области с левой стороны, а для изменения способов представления – пять кнопок, выстроенных вдоль правого края окна вверху. Если навести указатель мыши на каждую из кнопок и подождать несколько мгновений, будет выведена подсказка, поясняющая назначение той или иной кнопки. На рис. 8.28 продемонстрированы кнопки изменения спо-

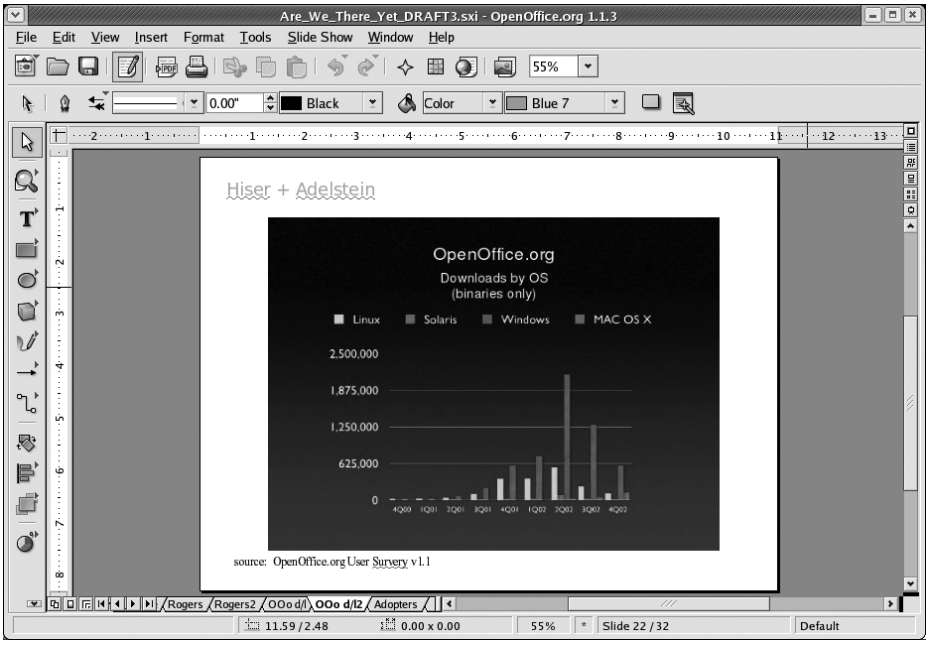


Рис. 8.29. *Управление режимами и способами представления*

собов представления, а на рис. 8.29 – кнопки управления и способами представления, и режимами.

Редактирование презентации

Изменение существующей презентации обычно ни у кого не вызывает затруднений.

Ввод текста. Чтобы ввести или отредактировать существующий текст, нужно щелкнуть на этом тексте. После щелчка вокруг текста появится затененная рамка с зелеными квадратиками на ней. Теперь можно переместить текстовый курсор в нужное место и внести изменения. Щелчок на любом другом месте слайда заставит затененную рамку исчезнуть.

Маркеры. Чтобы добавить в начало строки маркер, нужно щелкнуть на строке, в которую будет вставлен маркер, а затем на кнопке Bullets (Маркеры), расположенной в центре контекстной панели. В случае неуверенности в том, на какой кнопке щелкать, можно попробовать навести указатель мыши на кнопку, дождаться появления подсказки и выбрать ту, где появится подсказка Bullets (Маркеры).

Чтобы использовать маркеры другой формы, нужно щелкнуть на крайней правой кнопке (с изображением маркированного списка) в контекстной панели. После щелчка появится диалог, в котором можно выбрать наиболее подходящий тип маркеров, а также определить некоторые другие параметры форматирования.

Импорт графических изображений, таблиц и диаграмм. Чтобы импортировать графическое изображение, таблицу или диаграмму из другой программы, веб-

страницы или компонента OpenOffice, достаточно скопировать выбранный объект в буфер обмена в программе-источнике и вставить его в слайд.

Сделать это можно, например, следующим образом: выделить объект в окне программы-источника, нажать комбинацию Ctrl+C, чтобы скопировать его (то есть поместить объект в буфер обмена), затем щелкнуть на слайде и вставить скопированный объект нажатием комбинации Ctrl+V.

Добавление слайдов. Чтобы добавить или вставить слайд в презентацию, достаточно выбрать пункт главного меню Insert (Вставка)→Slide (Слайд), в появившемся диалоге Insert Slide (Вставка слайда) выбрать авторазметку (AutoLayout) и щелкнуть на кнопке OK.

Удаление слайдов. Чтобы быстро удалить слайд, нужно щелкнуть правой кнопкой мыши на вкладке этого слайда и выбрать пункт контекстного меню Delete (Удалить). Удалить слайд можно также с помощью главного меню, для чего нужно выбрать пункт Edit (Правка)→Delete Slide (Удалить слайд).

Перемещение слайдов. Самый простой способ переместить слайд из одного места презентации в другое состоит в том, чтобы просто щелкнуть на вкладке слайда и перетащить ее мышью на новое место среди других вкладок.

Панель презентации

Щелчок на правой крайней кнопке в контекстной панели открывает плавающую панель презентации, с помощью которой можно быстро выполнять некоторые действия во время работы над презентацией. В перечень функций, имеющихся на панели, входят: Insert Slide (Вставить слайд), Modify Slide Layout (Изменить разметку слайда), Slide Design (Дизайн слайда), Duplicate Slide (Дублировать слайд) и Expand Slide (Расширить слайд). Чтобы скрыть панель, нужно еще раз щелкнуть на правой крайней кнопке в контекстной панели.

Просмотр презентации

Просмотреть уже готовую презентацию – что может быть проще. Чтобы запустить показ презентации, нужно нажать клавишу F9, а чтобы закончить – клавишу Esc.

Смена слайдов. Чтобы определить эффекты при смене одиночного слайда, нужно выбрать пункт главного меню Slide Show (Демонстрация)→Slide Transition (Смена слайдов). На рис. 8.30 показаны различные эффекты при смене слайдов.

Имеется возможность выбирать низкую, среднюю или высокую скорость смены слайда в раскрывающемся списке в нижней части диалога Slide Transition (Смена слайда).

Если предполагается, что все слайды в презентации будут сменяться с одним и тем же эффектом, проще всего настроить эффект смены слайда для всех слайдов с помощью мастера при создании презентации.

Настройка порядка демонстрации. Существует возможность определить несколько версий одной и той же презентации, используя нужные слайды и разные варианты параметров настройки. Это удобно для адаптации отдельных частей презентации под определенную аудиторию. Эту возможность можно также использовать для предварительной настройки некоторых версий презентации, которые содержат более подробные сведения и которые можно переключать пря-

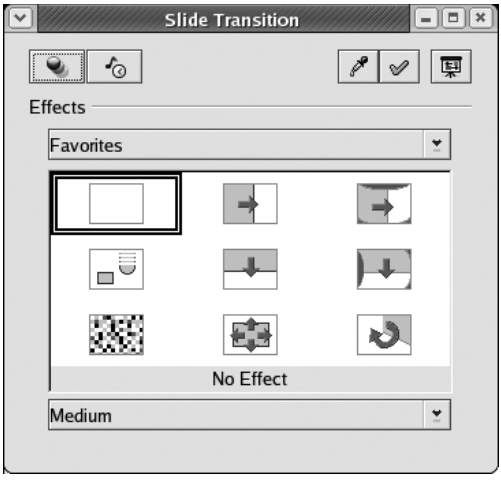


Рис. 8.30. *Диалог Slide Transition (Смена слайда)*

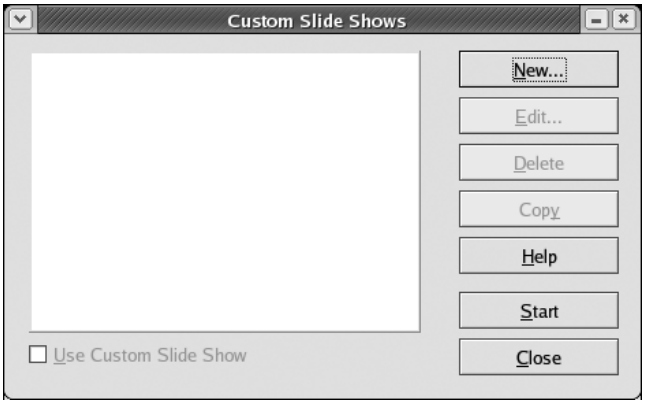


Рис. 8.31. *Диалог Custom Slide Show (Обычная демонстрация слайдов)*

мо по ходу демонстрации, чтобы скрыть некоторые сложности с целью экономии времени зрителей.

Чтобы определить новый вариант демонстрации слайдов, в главном меню следует выбрать пункт Slide Show (Демонстрация)→Custom Slide Show (Обычная демонстрация), после чего откроется диалог Custom Slide Show (Обычная демонстрация слайдов) (рис. 8.31), в котором следует щелкнуть на кнопке New (Создать).

После этого откроется диалог Define Custom Slide Show (Задать обычную демонстрацию слайдов) (рис. 8.32), в котором можно определить новую версию порядка демонстрации слайдов и выбрать, какие из слайдов следует включить в демонстрацию. Чтобы включить какой-либо слайд в демонстрацию, нужно сначала выделить его в списке Existing slides (Существующие слайды), расположенном в левой части диалога, а затем щелкнуть на верхней кнопке со стрелкой, и выбранный слайд появится в правом списке Selected slides (Выбранные слайды).

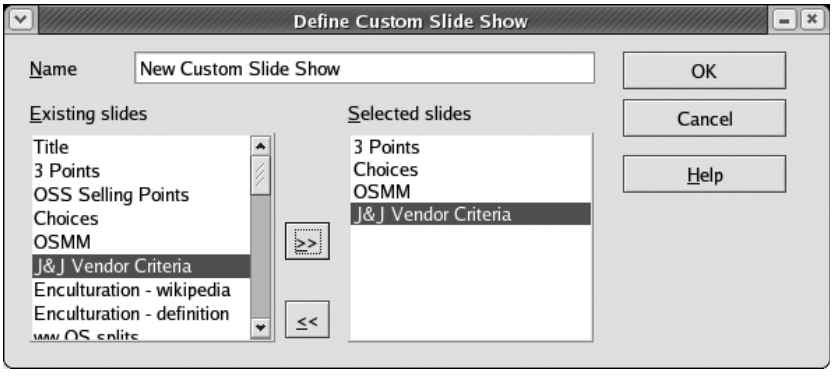


Рис. 8.32. *Диалог Define Custom Slide Show (Задать обычную демонстрацию слайдов)*

Настройка OpenOffice

Потратив немного времени на настройку нескольких параметров, отвечающих вашим потребностям, можно сэкономить уйму времени.

Добавление объектов запуска

Существует возможность создания объектов запуска (ярлыков на рабочем столе или кнопок на панелях) как самого OpenOffice без запуска какого-либо конкретного компонента, так и отдельных компонентов пакета OpenOffice.

Процедура добавления объекта запуска OOoWriter на рабочий стол уже была описана в разделе «Добавление ярлыка OOoWriter на рабочий стол» выше в этой главе. Добавление элементов запуска других компонентов, таких как OOoCalc или OOoImpress, выполняется аналогичным образом.

Выбор формата MS Office по умолчанию при сохранении файлов

Чтобы вынудить OOoWriter автоматически сохранять файлы документов в формате MS Word (*.doc*), нужно выбрать пункт главного меню Tools (Сервис)→Options (Параметры) и в диалоге Options (Параметры) выбрать пункт Load/Save (Загрузка/сохранение)→General (Общие). В правой части диалога откроется страница Option (Параметры) – Load/Save (Загрузка/Сохранение) – General (Общие). На этой странице в поле Document type (Тип документа) раздела Standard file format (Формат файла по умолчанию) уже будет выбран пункт Text document (Текстовый документ). Его нужно оставить как есть, а в раскрывающемся списке Always save as (Всегда сохранять как), расположенном правее, выбрать одну из трех версий MS Word:

- Microsoft Word 6.0.
- Microsoft Word 95.
- Microsoft Word 97/2000/XP.

После этого можно щелкнуть на кнопке OK. Необходимо проявлять большую осторожность при выборе версии формата MS Word. На сегодняшний день версия Microsoft Word 97/2000/XP используется подавляющим большинством, однако

если среди людей, окружающих вас, есть такие, кто пользуется более старыми версиями (6.0 или 95), то, очевидно, выбор должен пасть на более ранние версии.

KOffice

OpenOffice – не единственный доступный в Linux пакет офисных приложений, распространяемый с открытыми исходными текстами. В рамках проекта KDE также был создан полноценный офисный пакет под названием KOffice, который придерживается устоявшихся стандартов и прекрасно вписывается в окружение рабочего стола KDE.[1]

KOffice представляет собой высокоинтегрированный пакет офисных приложений, построенных непосредственно на технологиях, применяемых в KDE. Это дает определенные преимущества в смысле интеграции, функциональных возможностей, производительности, знакомого внешнего вида и т. д. Таким образом, KOffice может извлекать выгоду из всех новейших технологий, используемых в KDE, таких как DCOP, KIO и KParts. В частности, технология KParts была расширена специально для компонентов KOffice, чтобы обеспечить гибкую возможность встраивания одних документов в другие. Компоненты KOffice прекрасно интегрируются друг в друга. Благодаря этому обычная электронная таблица может содержать в себе элементы от диаграмм до целых презентаций, отчетов и даже текстовых документов. Аналогичным образом, практически любой компонент может содержать в себе любой другой. Компоненты полностью встраиваемы, что дает пользователю возможность выполнять любые операции, которые доступны в автономных приложениях.

Поскольку многие технологии уже имеют реализацию непосредственно внутри KDE, пакет KOffice получился очень легким, что выражается в уменьшении времени запуска приложений и сниженном потреблении памяти. Все это делает KOffice весьма привлекательным пакетом офисных приложений, способным работать даже на устаревших компьютерах, что в некоторых случаях может помочь сэкономить немало денег.

Пакет KOffice обладает весьма широкими возможностями. В его состав входят не только текстовый процессор, электронная таблица и приложение для создания и просмотра презентаций, но и другие компоненты: графический редактор, редактор блок-схем, генератор отчетов, средства управления базами данных и приложение для управления проектами. Благодаря высокой гибкости и тесной интеграции с KDE такие утилиты, как механизм диаграмм и схем, а также редактор формул, доступны в виде самостоятельных приложений. Простой KDE-подобный внешний вид и удобный интерфейс делают KOffice подходящим инструментом для повседневной работы в офисе.

Офисный пакет KOffice очень велик, чтобы подробно описывать каждую его особенность. Наиболее общие из них включают абстракцию местоположения документа, управление приложениями средствами DCOP, внедрение документов и поддержку модулей расширений. Посетите веб-сайт KOffice (*http://www.koffice.org*), чтобы узнать о последних достижениях пакета.

---

[1] Этот раздел был написан Рафаэлем Лангехостом (Raphael Langerhorst), членом команды документирования KDE.

Приложения пакета KOffice поддерживают стандарт формата файлов OASIS OpenDocument, благодаря чему возможна работа с одними и теми же документами с помощью других программ, поддерживающих этот же стандарт, как, например, OpenOffice.

Для офисного пакета чрезвычайно важно следовать стандартам везде, где только возможно, и в особенности это относится к формату файлов. Благодаря этому пользователи могут быть уверены, что их документы будут открываться и в далеком будущем, независимо от того, по какому пути пойдет развитие нынешнего программного обеспечения. Спецификация формата файлов OASIS OpenDocument – это открытый стандарт для офисных приложений. Этот формат используется как в KOffice, так и в OpenOffice, то есть файлы могут беспрепятственно курсировать между двумя пакетами.

В состав KOffice входит больше компонентов, чем определено спецификациями OASIS. Однако те компоненты, которые подпадают по действие этих спецификаций, в качестве стандартного используют формат файлов OASIS OpenDocument.

Ниже приводится перечень основных компонентов KOffice:

Текстовый процессор и настольная издательская система: KWord

KWord – это текстовый процессор, но он обладает многими характеристиками, присущими настольным издательским системам. Разнообразные возможности KWord позволяют создавать интересные варианты компоновки документа.

Электронная таблица: KSpread

KSpread – типичное приложение электронной таблицы. Оно предлагает большое число функций, вариантов форматирования, возможность создания книг из множества листов, диаграммы, схемы и многое другое. Разумеется, оно легко может интегрироваться с любым другим приложением KOffice для расширения своих возможностей.

Презентации: KPresenter

KPresenter – это компонент презентаций. Он может использоваться для создания экранных презентаций или для создания и печати транспарантов.

Блок-схемы: Kivio

Компонент Kivio может использоваться для создания любого рода блок-схем и диаграмм. Для Kivio существуют специализированные комплекты трафаретов. С его помощью возможно создавать даже диаграммы UML.

Векторная графика: Karbon14

Karbon14 – это редактор векторной графики.

Пиксельная графика: Krita

Krita – это инструмент для создания высококачественных пиксельных графических изображений. Обладает большим числом модулей расширения для работы с графическими изображениями и поддерживает различные форматы файлов.

Отчеты: Kugar

Kugar может использоваться для создания отчетов бизнес-класса, которые создаются с помощью дизайнера отчетов.

Управление базами данных и формами: Kexi

Kexi – полнофункциональное приложение управления базами данных. В нем можно проектировать формы для работы с данными. Поддерживает самые разнообразные типы баз данных, например PostgreSQL или MySQL. Кроме того, имеется возможность импортировать файлы MS Access (*.mdb*).

Каждый компонент KOffice идет в комплекте со своим справочным руководством. Эти руководства содержат самые последние сведения о различных компонентах KOffice и потому рекомендуются к прочтению, чтобы узнать о KOffice еще больше. Немалое количество информации о KOffice можно найти на веб-сайте проекта.

Основной сайт в Интернете находится по адресу: *http://www.koffice.org*. Здесь можно найти сведения о списках рассылки для пользователей и разработчиков, а также ссылки на дополнительные ресурсы для разработчиков.

Проект Kexi имеет дополнительный веб-сайт *http://www.kexiproject. org*.

В следующих двух разделах мы не будем описывать стандартные возможности пакета, а займемся более подробным исследованием двух интересных особенностей в надежде подтолкнуть вас тем самым к дальнейшему самостоятельному изучению KOffice.

Из рук в руки: сведения о KOffice

Сейчас мы разберем некоторые примеры работы с KOffice, чтобы узнать о нем чуть больше. Будет неплохо, если по ходу обсуждения вы сами попробуете создавать документы и самостоятельно экспериментировать с ними. Цель этих примеров состоит в том, чтобы дать вам почувствовать KOffice, не пытаясь при этом провести полный критический анализ, для которого пришлось бы написать целую книгу. Не забывайте, справочные руководства, включенные в состав KOffice, содержат большое число подробных сведений, которые позволят вам изучить все имеющиеся компоненты.

Табуляторы в KWord

Табуляторы используются для выравнивания текста по горизонтали, а также для создания простейших таблиц или списков, в которых отдельные части текста должны располагаться с определенным интервалом по горизонтали. Табуляторами можно пользоваться и просто для того, чтобы сместить текст в определенную позицию по горизонтали.

В KWord табуляторы обладают различными характеристиками. Они допускают установку различных способов выравнивания текста, например: по левому краю, по правому краю, по центру или по определенному символу (например, по запятой). В частности, последний тип выравнивания можно использовать для составления списков с дробными числами или ценами.

Табуляторы – это часть формата абзаца. Таким образом, все, что связано с табуляторами, можно настроить в диалоге Paragraph Settings (Свойства абзаца) (рис. 8.33). Чтобы открыть этот диалог, нужно выбрать пункт главного меню Format (Формат)→Paragraph (Абзац).

Сейчас мы создадим несколько табуляторов. Мы начнем редактировать табуляторы простым интуитивным способом, а позднее коснемся некоторых деталей настройки.

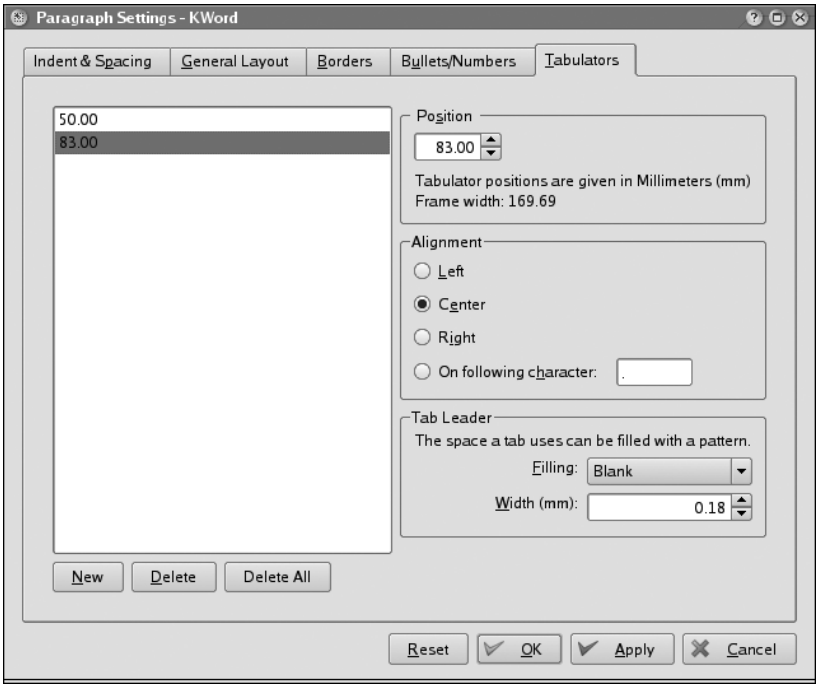


Рис. 8.33. *Настройка табуляторов в KWord*

Запустите KWord, выберите шаблон U. S. letter (Чистый лист) и щелкните на кнопке OK (рис. 8.34).

Теперь взгляните на верхнюю линейку документа (рис. 8.35). Белая область на верхней линейке соответствует ширине области редактирования документа. Эта часть линейки может использоваться для установки табуляторов. В левом верхнем углу можно наблюдать маленький значок, который представляет текущий тип табулятора. Как уже говорилось выше, табуляторы могут иметь выравнивание по левому краю, по правому краю, по центру или по определенному символу. Изменить тип табулятора можно простым щелчком мыши на этом значке. Посмотрите, как меняется изображение значка при смене типа табулятора.

Далее вы должны установить несколько табуляторов, указав их типы и расставив в нужные позиции на верхней линейке (рис. 8.36).

Прежде всего, необходимо установить тип табулятора, щелкнув на значке в левом верхнем углу, а затем щелкнув левой кнопкой мыши в нужном месте на линейке. Добавьте четыре различных типа табуляторов, чтобы верхняя линейка выглядела у вас примерно так, как показано на рис. 8.36.

Чтобы увидеть, как действуют эти табуляторы, просто введите какой-нибудь текст в каждую из позиций:

1. Нажмите клавишу табуляции один раз. После этого текстовый курсор должен переместиться в позицию первого табулятора.

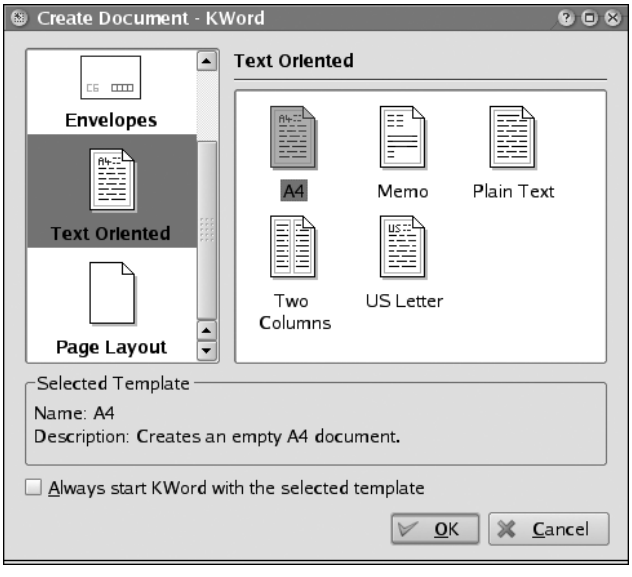


Рис. 8.34. *Выбор шаблона текстового документа*

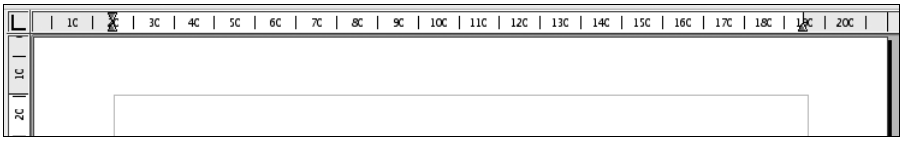


Рис. 8.35. *Верхняя линейка KWord*

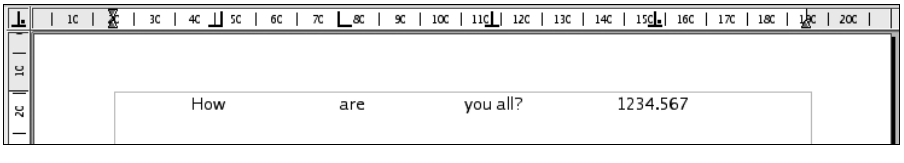


Рис. 8.36. *Линейка с установленными табуляторами*

2. Введите несколько символов, например слово `How`. Текст должен выровняться по правому краю, это свидетельствует о том, что табулятор имеет тип с выравниванием по правому краю.
3. Нажмите клавишу табуляции еще раз. Теперь текстовый курсор должен находиться в позиции второго табулятора.
4. Введите еще одно слово, например `are`. Текст должен выровняться по левому краю, это свидетельствует о том, что табулятор имеет тип с выравниванием по левому краю.
5. Нажмите клавишу табуляции еще раз. Теперь текстовый курсор должен находиться в позиции третьего табулятора.

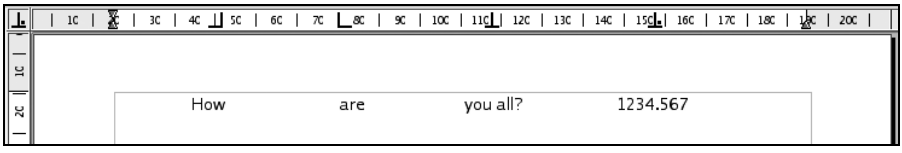


Рис. 8.37. *Элементы текста расставлены по табуляторам*

6. Введите еще какой-нибудь текст, например `you all?`. Текст должен выровняться по центру; это свидетельствует о том, что третий табулятор имеет тип с выравниванием по центру.
7. Нажмите клавишу табуляции еще раз, чтобы перейти в позицию четвертого (последнего) табулятора.
8. Введите число, например `12345.234`. Обратите внимание: число должно выровняться по десятичной точке. Этот тип табуляторов очень удобно использовать для работы с числами.

Текст должен выглядеть примерно так, как показано на рис. 8.37.

Другие параметры табуляторов можно настроить с помощью диалога Paragraph Settings (Свойства абзаца). Проще всего вызвать этот диалог двойным щелчком на одном из табуляторов в верхней линейке. В результате на экране появится диалог, который уже упоминался в начале этого раздела. До дополнительных параметров настройки можно также добраться, выбрав пункт Paragraph (Абзац) в меню Format (Формат) и затем перейдя на вкладку Tabulators (Позиции табуляции). Попробуйте поэкспериментировать с параметрами настройки, их назначение вполне очевидно. В этом же диалоге можно добавлять и удалять табуляторы.

Если для определенного стиля необходимы постоянные табуляторы, то тогда для расстановки табуляторов следует воспользоваться менеджером стилей, доступ к которому осуществляется также через меню Format (Формат).

Встраивание диаграмм в электронные таблицы

Компонент электронных таблиц, входящий в состав KOffice, называется Kspread. Помимо различных вычислений он предоставляет возможности создания диаграмм для визуализации данных.

Для визуализации данных KSpread использует компонент KChart. Кроме того, KChart может использоваться как самостоятельное приложение для создания разнообразных диаграмм.

Теперь мы рассмотрим, как создаются простейшие диаграммы внутри KSpread.

Предположим, что вы являетесь работником компании, которая имеет дело с несколькими продуктами питания, и вам необходимо увидеть, какой объем прибыли приносит каждый из них, и сравнить результаты. В этом вам поможет обычная диаграмма.

Запустите KSpread с пустой таблицей. Внешний вид окна приложения должен напоминать рис. 8.38.

Ввод данных в таблицу выполняется достаточно просто: нужно просто перейти в требуемую ячейку с помощью клавиш управления курсором или щелкнуть на

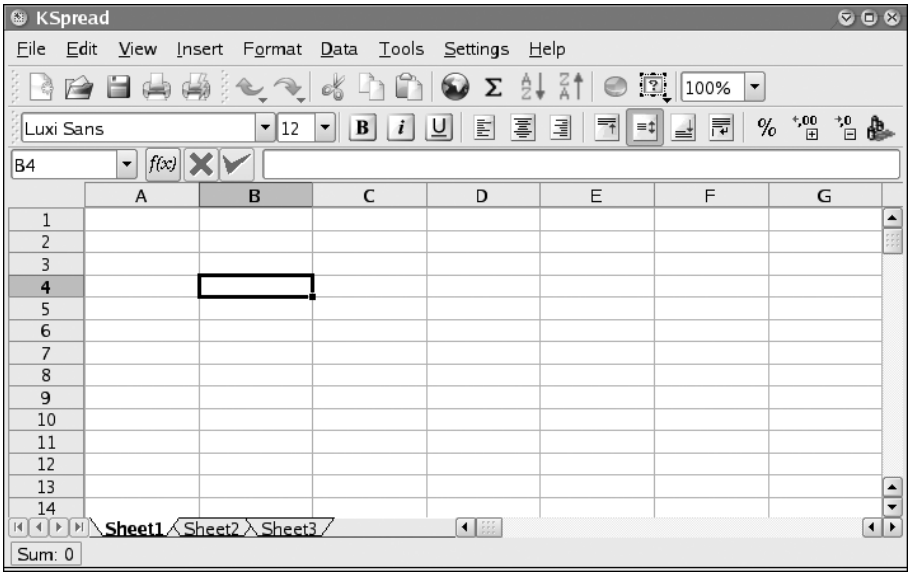


Рис. 8.38. *Приложение KSpread сразу после запуска*

	Oranges	Bread	Bananas	Apples
Expense	200	300	200	320
Income	250	480	340	350
Profit	50	180	140	30

Рис. 8.39. *Данные в электронной таблице*

ячейке мышью. Теперь введите данные в таблицу, как показано на рис. 8.39. Совершенно не имеет значения, с какой ячейки будет начинаться ввод. В этом примере мы ввели слово Expense (Затраты) в ячейку B4. По окончании ввода данных выделите область, чтобы из нее создать диаграмму, как показано на рис. 8.40.

Теперь нужно щелкнуть на кнопке Insert Chart (Вставить диаграмму). Изображение указателя мыши приобретет вид крестика, это говорит о том, что нужно выделить область, куда можно будет вставить диаграмму. Просто нарисуйте прямоугольник левой кнопкой мыши под таблицей с данными. После того как кнопка мыши будет отпущена, мастер создания диаграмм попросит выбрать тип диаграммы. Оставьте выделенным по умолчанию тип Bar (Гистограмма) и щелкните на кнопке Finish (Готово) (рис. 8.41).

Результат должен выглядеть так, как показано на рис. 8.42. В этой диаграмме можно увидеть объемы затрат (столбики красного цвета, хотя в книге изображение напечатано в черно-белом виде), выручки (зеленый) и прибыли (синий) для каждого из продуктов. Если необходимо увидеть соотношения в процентах, просто щелкните на диаграмме дважды.

Обратите внимание на изменения, произошедшие в меню и панелях инструментов. Это наглядный пример тесной интеграции компонентов, входящих в состав

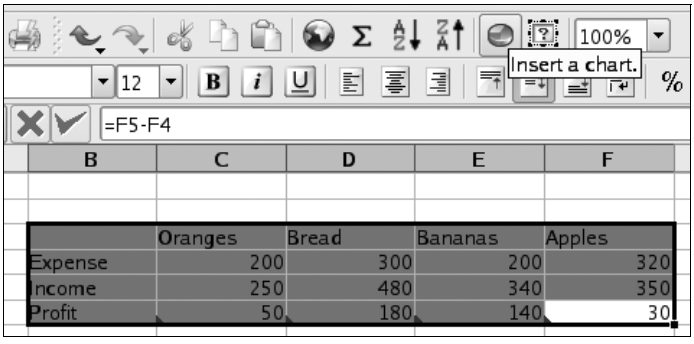


Рис. 8.40. *Выбор области с данными, на основе которых будет создана диаграмма*

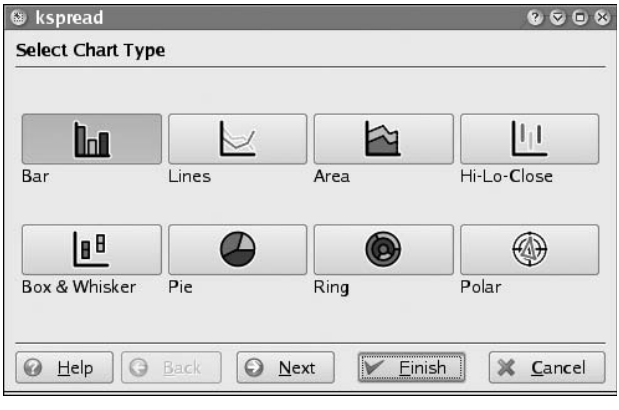


Рис. 8.41. *Мастер создания диаграмм*

KOffice, в результате которой имеется возможность использовать одни компоненты внутри других совершенно прозрачным образом. В данном случае меню и панели инструментов относятся к компоненту KChart.

Теперь нужно щелкнуть правой кнопкой мыши на диаграмме и выбрать в контекстном меню пункт Configure Chart (Редактировать диаграмму). После этого на экране появится диалог настройки диаграммы, где следует перейти на вкладку Chart Subtype (Подтип диаграммы) (рис. 8.43). На этой вкладке можно выбрать один из предлагаемых подтипов для текущей диаграммы.

Для данного случая просто выберите параметр Percent (Процентный) и щелкните на кнопке OK. Конечный результат, получившийся у нас, приводится на рис. 8.44. Каждый из продуктов приводится к шкале 100%, и мы сразу можем видеть, сколько было затрачено на каждый из продуктов и сколько выручки принес каждый из них, а прибыль показывает разницу между этими двумя величинами. Исходя из наших данных, можно сделать вывод, что наибольшую выгоду приносят бананы, тогда как яблоки – самую малую.

А теперь двигайтесь дальше самостоятельно, попробуйте различные варианты диаграмм и посмотрите, как еще могут быть представлены одни и те же данные!

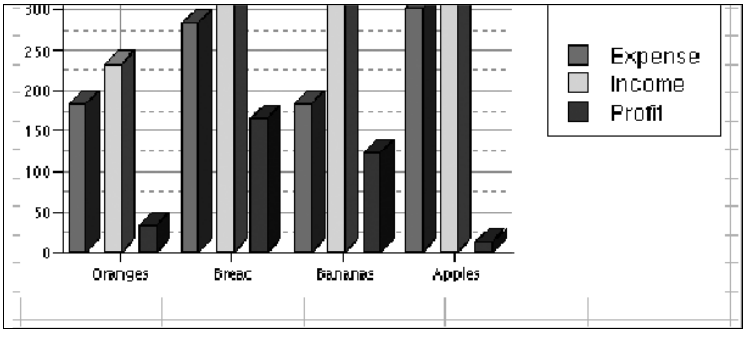


Рис. 8.42. *Получившаяся диаграмма*

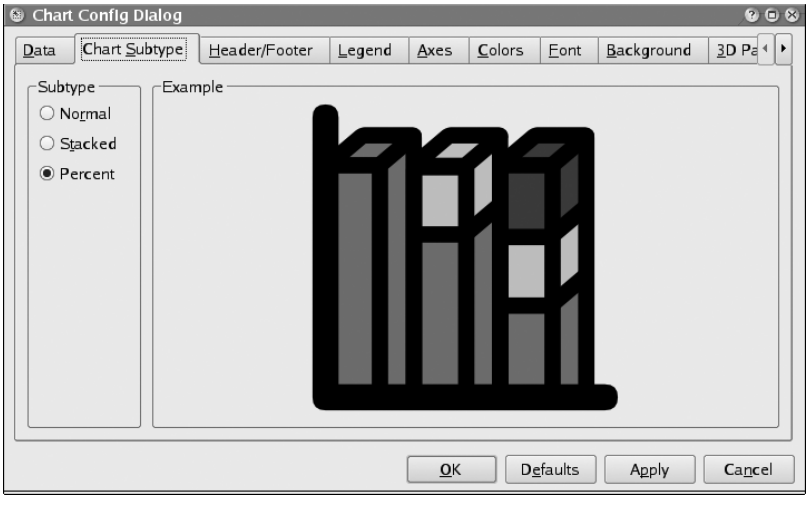


Рис. 8.43. *Выбор подтипа диаграммы*

Другие текстовые процессоры

Хотя текстовые процессоры, обсуждавшиеся выше, пока являются самыми популярными среди пользователей Linux, эта книга не была бы полной, если бы мы не упомянули некоторые альтернативы.

Anyware Office от компании VistaSource, Inc.

Anyware Office – это пакет офисных приложений для Linux, хотя и коммерческий, но достаточно недорогой. Он включает в себя не только текстовый процессор, но и электронную таблицу, графический редактор, программу чтения электронной почты и ряд других инструментальных средств. Поведение приложений из пакета Anyware Office несколько отличается от поведения текстовых процессоров типа Microsoft Word или WordPerfect, но после привыкания к некоторым его особенностям работать с ним становится легко и просто. Самая примечательная особенность – возможность импортировать

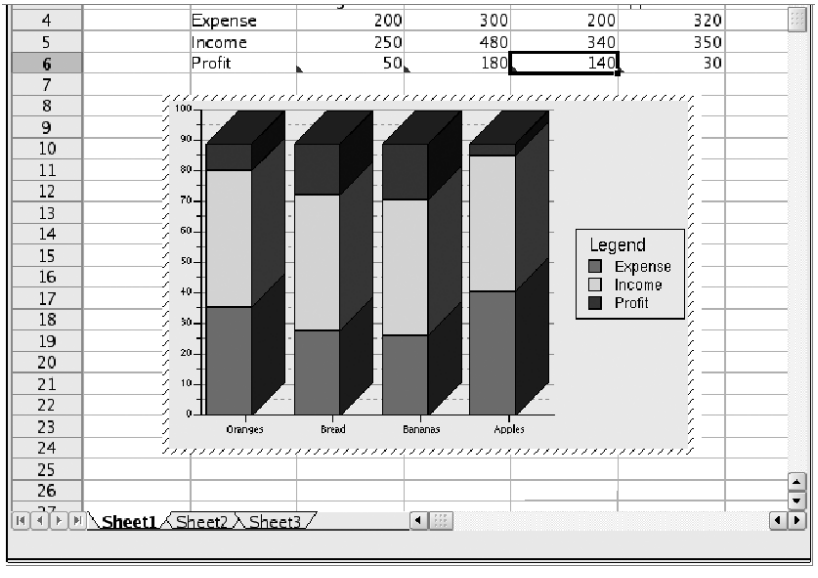


Рис. 8.44. *Окончательный вариант диаграммы*

и экспортировать документы FrameMaker. Однако похоже, что работа над этим проектом прекратилась и будущее его пока остается неясным.

AbiWord

Дополнительную информацию об этом текстовом процессоре можно найти на сайте *http://www.abiword.org*.

LyX

Пакет LyX (имеющийся также в варианте KLyX с более современным интерфейсом пользователя) обеспечивает интерфейс пользователя WYSIWYG, работающий с оконными менеджерами из стандартных дистрибутивов Linux и использующий при форматировании текста для печати пакеты $LaTeX$ и TeX. Если вы можете смириться с ограничениями форматирования, накладываемыми этим пакетом (а большинство из нас может), то LyX/KLyX будет для вас отличным решением. LyX/KlyX не может отображать некоторые мощные функции форматирования, обеспечиваемые TeX, поэтому если вы усиленно используете TeX, то это решение не для вас. В большинство дистрибутивов Linux LyX/KlyX не входит. Для работы с ним вам придется загрузить его из архива Linux.

Синхронизация с PDA

Карманные компьютеры (Personal Digital Assistant, PDA) стали в наше время вполне обычным явлением, и поклонники Linux хотели бы использовать их со своей любимой операционной системой. В этом разделе мы расскажем, как синхронизировать карманные компьютеры с настольными компьютерами, работающими под управлением Linux.

В этом разделе рассказывается не о запуске Linux на карманных компьютерах, хотя в принципе это возможно. Многие благополучно запускают Linux на карманных компьютерах линейки Hewlett-Packard/Compaq iPaq и используют прикладное программное обеспечение для этой операционной системы. А серия карманных компьютеров Sharp Zaurus даже поставляется с предустановленной ОС Linux, хотя при работе с устройством это не так очевидно. Большой объем информации о запуске Linux на PDA можно найти на сайте *http://www.handhelds.org*.

Пользование карманным компьютером совместно с настольным предполагает необходимость синхронизации данных в карманном компьютере с данными в настольном компьютере. Например, любой пользователь наверняка захотел бы иметь одну и ту же адресную книгу на обоих компьютерах, и в этом ему поможет программное обеспечение для выполнения синхронизации.

Не следует ожидать, что производители или поставщики карманных компьютеров будут распространять программное обеспечение синхронизации для Linux. Даже Sharp Zaurus, на котором, как упоминалось выше, предустановлена ОС Linux, поставляется с программным обеспечением синхронизации для ОС Windows. Но, как обычно, пользователи Linux оказались в состоянии создать свое собственное программное обеспечение.

Процедура синхронизации карманного компьютера с настольным связана с выполнением следующих действий:

- Подключение карманного компьютера и распознавание его в операционной системе Linux.
- Установка программного обеспечения обслуживания специализированного оборудования для синхронизации, как, например, кнопки HotSync.
- Установка программного обеспечения, выполняющего собственно синхронизацию объектов данных.
- Использование программного обеспечения, которое гарантирует выполнение синхронизации на прикладном уровне (то есть между календарем PDA и календарем в настольном компьютере).

Проверка соединения

Сначала определимся с аппаратной частью. Карманные компьютеры обычно соединяются с настольным компьютером посредством подставки – маленького модуля, который подключен к компьютеру с помощью провода и имеет контакты для подключения PDA. Иногда карманный и настольный компьютеры соединяются специализированным кабелем напрямую. Подключение к настольному компьютеру обычно осуществляется через интерфейс USB, реже – через последовательный интерфейс.

В первую очередь, чтобы получить работающее соединение, ядро настольной системы должно видеть и правильно распознавать PDA. Попробуйте подключить PDA (вставить его в подставку или воспользоваться кабелем прямого соединения) к настольному компьютеру. Затем посмотрите, какие сообщения появятся в журнале ядра, для чего нужно зарегистрироваться как пользователь *root* и ввести команду `tail -f /var/log/messages`. (Подробнее о журналируемых сообщениях ядра будет рассказываться в разделе «Управление системными журналами» главы 10.)

Теперь, не прекращая следить за изменениями в системном журнале, инициируйте синхронизацию со стороны карманного компьютера, например, нажав кнопку HotSync на подставке или запустив на карманном компьютере команду синхронизации. Если PDA подключен через интерфейс USB, в журнале должны появиться строки примерно следующего содержания (некоторые строки были сокращены, чтобы уместить их на книжную страницу):

```
Jun 21 10:32:52 tigger kernel: ohci_hcd 0000:02:06.1: wakeup
Jun 21 10:32:52 tigger kernel: klogd 1.4.1, ---------- state change ----------
Jun 21 10:32:52 tigger kernel: usb 3-2: new full speed USB device using addr
Jun 21 10:32:52 tigger kernel: usb 3-2: Product: Palm Handheld
Jun 21 10:32:52 tigger kernel: usb 3-2: Manufacturer: Palm, Inc.
Jun 21 10:32:52 tigger kernel: usb 3-2: SerialNumber:3030063041944034303506909
Jun 21 10:32:52 tigger kernel: visor 3-2:1.0: Handspring Visor/Palm OS convert
Jun 21 10:32:52 tigger kernel: usb 3-2: Handspring Visor/Palm OS converter now
Jun 21 10:32:52 tigger kernel: usb 3-2: Handspring Visor/Palm OS converter now
```

В данном примере ядро обнаружило подключение карманного компьютера Palm Tungsten T3 через интерфейс USB. Если ничего обнаружено не было, причины могли быть следующими: обрыв соединяющего кабеля, возможно, запрос на синхронизацию не был распознан, в ядре могут отсутствовать необходимые модули драйвера. В главе 18 подробно рассказывается о модулях драйверов устройств и их установке в ядро.

Синхронизация с помощью KPilot

После того как соединение будет установлено, вам потребуется программное обеспечение, которое будет выполнять синхронизацию данных через соединение. Для обширного семейства карманных компьютеров Palm (куда входят Sony Clie, Handspring Visor и множество других совместимых с ними моделей) таким программным обеспечением является пакет *pilot-link*. Этот пакет уже включен во многие популярные дистрибутивы; если в вашем дистрибутиве его нет, пакет можно загрузить с сайта *http://www.pilot-link.org*. Программы из этого пакета обычно не используются напрямую – чаще всего они вызываются через другое прикладное программное обеспечение, которое основывается на них.

Помимо стандартных блоков, на основе которых строится программное обеспечение синхронизации, в пакет входят небольшие приложения – *каналы* (*conduits*), каждое из которых поддерживает синхронизацию данных определенного типа. Существуют каналы для календаря, адресной книги и т. д.

До этого момента программное обеспечение и выполняемые действия, описанные выше, зависели от типа карманного компьютера, который необходимо синхронизировать, и совершенно не зависели от программного окружения рабочего стола. Программное обеспечение, которое может использоваться для синхронизации, отличается для разных рабочих столов. Здесь мы рассмотрим KPilot – достаточно крупный программный пакет для рабочего стола KDE, который выполняет синхронизацию Palm-подобных PDA с такими приложениями рабочего стола KDE, как KOrganizer и KAddressBook, а также с приложениями рабочего стола GNOME, как, например, Evolution.

Пакет KPilot (сайт проекта: *http://www.kpilot.org*) содержит две программы: *kpilotDaemon* и *kpilot*. Теоретически для нужд синхронизации необходима только

программа *kpilotDaemon* (она ожидает нажатия кнопки HotSync и затем выполняет синхронизацию). Но на практике многие предпочитают пользоваться программой *kpilot*, по крайней мере, на начальном этапе. Эта программа позволяет выполнять настройки демона *kpilotDaemon* и контролировать процесс синхронизации.

После запуска KPilot (рис. 8.45) нужно выбрать пункт главного меню Settings (Настройка)→Configure KPilot (Настроить KPilot). Когда программа предложит запустить мастер настройки, щелкните на кнопке. На первой странице потребуется указать имя пользователя карманного компьютера (чтобы синхронизация была выполнена с нужными данными) и порт настольного компьютера, к которому подключен карманный компьютер. KPilot предложит выполнить определение порта автоматически, и этим предложением не стоит пренебрегать. Если автоматическое определение подключения не даст желаемого результата (но при этом фактическое подключение аппаратных средств работает, как было описано в предыдущем разделе), попробуйте указать устройство */dev/ttyUSB1* или */dev/ttyUSB2* (или порты с более высокими порядковыми номерами) в случае подключения PDA к порту USB и */dev/ttyS0* или */dev/ttyS1* в случае подключения

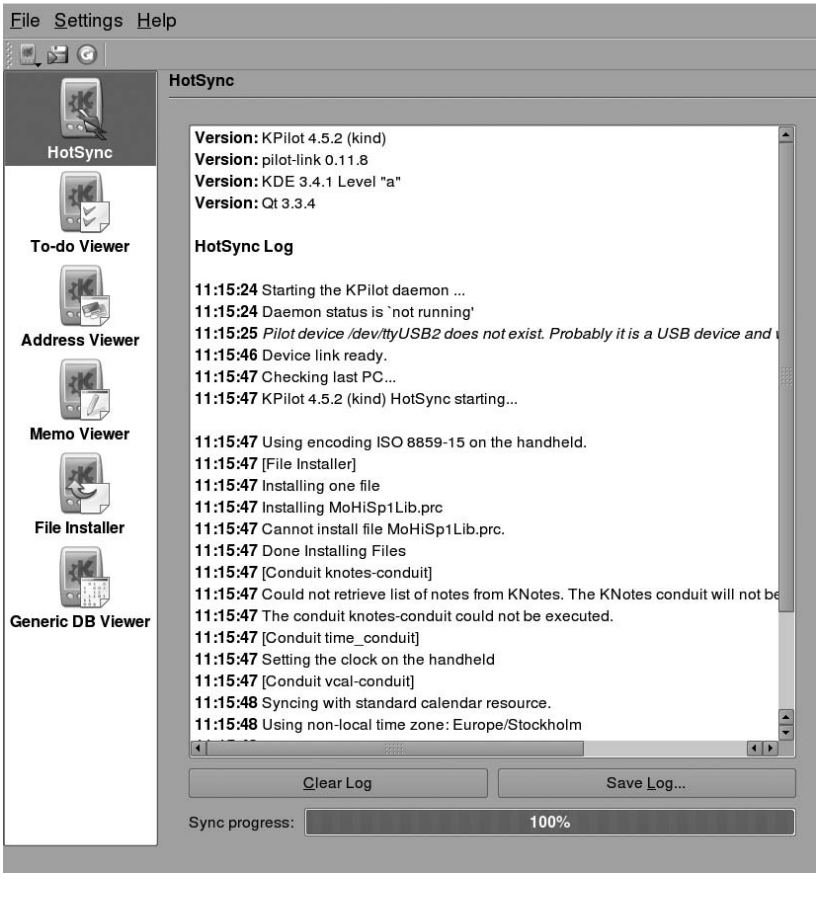


***Рис. 8.45.** Синхронизация с помощью KPilot*

к последовательному порту. На следующей странице необходимо будет указать, какие приложения рабочего стола должны быть синхронизированы.

Указав правильный набор, можно дать приложению KPilot возможность попытаться выполнить синхронизацию. В результате автоматически запустится приложение *kpilotDaemon*, если оно еще не было запущено.

При выполнении следующих действий наблюдайте за изменением содержимого окна HotSync Log в KPilot – здесь может появиться важная информация, которая поможет в устранении возможных проблем. Если вы увидите сообщение «Pilot device /dev/USB2 does not exists. Probably it is a USB device and will appear during a HotSync» («Устройство /dev/USB2 не найдено. Вероятно, это устройство USB, которое будет обнаружено после нажатия кнопки HotSync») или нечто подобное, можете не волноваться.

Теперь нажмите кнопку HotSync на подставке или запустите синхронизацию каким-либо способом со стороны PDA. Если вы увидите сообщение «Device link ready» («Соединение с устройством установлено») плюс еще целую серию сообщений, информирующих о ходе выполнения синхронизации по различным каналам, значит, все идет так, как надо. Обратите внимание: если на карманном компьютере установлено достаточно много приложений, синхронизация может занять значительное время.

Чего следует ожидать при работе с операционной системой Linux? Синхронизация со стандартными приложениями, такими как календарь, адресная книга и заметки, должна работать без каких-либо проблем. Для многих других коммерческих приложений, имеющихся в карманном компьютере, в Linux нет аналогов, но, так как KPilot в состоянии синхронизировать базы данных Palm, фактически никак не анализируя их содержимое, можно, по крайней мере, копировать и восстанавливать эти данные. Можно также устанавливать пакеты прикладных программ с помощью инсталлятора файлов KPilot. Даже популярное программное обеспечение синхронизации канала новостей AvantGo прекрасно работает в Linux.

Обычно не работают (или работают, но требуют от вас существенных усилий): доступ к дополнительным носителям данных, таким как карты CompactFlash, и приложения, которые реализуют дополнительные функциональные возможности синхронизации (например, загрузка новых баз данных с веб-сайта как часть процесса синхронизации). Типичный пример последней категории – приложения с расписанием движения самолетов. Таким образом, если в вашем распоряжении имеется компьютер с операционной системой Windows (или на вашем компьютере установлены две операционных системы – Linux и Windows), возможно, будет лучше по-прежнему использовать программное обеспечение Windows для выполнения синхронизации. Для повседневных нужд Linux и PDA (по крайней мере, Palm-подобный PDA) являют собой замечательную комбинацию.

В настоящее время ведутся работы по созданию унифицированного приложения синхронизации, получившего название KitchenSync. Как только оно будет готово, им можно будет заменить не только KPilot, но и другие пакеты синхронизации с карманным компьютером, а также множество более мелких пакетов, предназначенных для синхронизации настольного компьютера с различными типами сотовых телефонов. Подробную информацию о ходе разработки KitchenSync

можно найти на сайте *http://www.handhelds.org/~zecke/kitchensync.html*. Еще одна программа, преследующая ту же цель, – OpenSync.

Программное обеспечение для рабочих групп

Одна из редко используемых возможностей, которыми обладают компьютеры, – это оказание помощи группе людей в координации их работы или личной жизни – составлении расписаний, списков задач, заметок, адресных книг и всего того, что помогает решать реальные повседневные проблемы. Представьте себе, что встречу можно перенести, просто перетащив текстовое поле в другое место в приложении календаря, а программная система автоматически отправит изменения всем, кого это касается, попросит подтвердить свое желание встретиться и автоматически обновит их расписание. Такое программное обеспечение, которое поддерживает группы людей, работающих в коллективе и координирующих свои действия друг с другом, обычно называется *программным обеспечением для рабочих групп*.

Для любых групп, кроме очень маленьких, информацию общего пользования членов коллектива обычно принято хранить в одном месте на сетевом ресурсе. Часто для этих целей выделяется отдельный компьютер, который называется *сервер рабочих групп* (*groupware server*). Доступ к этому серверу осуществляется разными способами и с помощью различного программного обеспечения для рабочих групп. В большинстве случаев доступ к серверу выполняется с помощью веб-броузеров. Во многих случаях используются полноценные приложения-клиенты, такие как Kontact или Evolution, которые подключаются к серверу с помощью различных протоколов чтения и управления данными. Поэтому подобные приложения часто упоминаются как *пакеты программ для рабочих групп*.

Сначала мы рассмотрим, какими возможностями обладают клиенты без организации доступа к серверу, а затем исследуем различные доступные серверные решения и их сильные стороны.

Основы организации групп

Благодаря набору имеющихся стандартов Интернета пользователи программного обеспечения для рабочих групп могут организовать свое сотрудничество не только с использованием сервера своей группы, расположенного внутри организации, но и до определенной степени со своими партнерами, использующими различное клиентское программное обеспечение для рабочих групп и серверы, работающие под управлением Linux или Windows. Делается это посредством отправки сообщений электронной почты, которые в качестве вложений содержат дополнительную информацию, воспринимаемую программным обеспечением для рабочих групп. Все пакеты программ для рабочих групп, доступные в Linux (Kontact, Evolution и Mozilla), поддерживают такую возможность, так же как и коммерческие клиенты для Windows и Mac OS (например, MS Outlook или Lotus Notes).

В качестве примера рассмотрим, что произойдет, когда вы пригласите вашего коллегу, до сих пор работающего с Windows и использующего MS Outlook, прийти к вам в среду на барбекю. Для этого нужно открыть свой календарь с расписанием на текущую неделю и создать новое событие в среду днем (рис. 8.46, в данном примере используется приложение Kontact). Далее необходимо добавить кол-

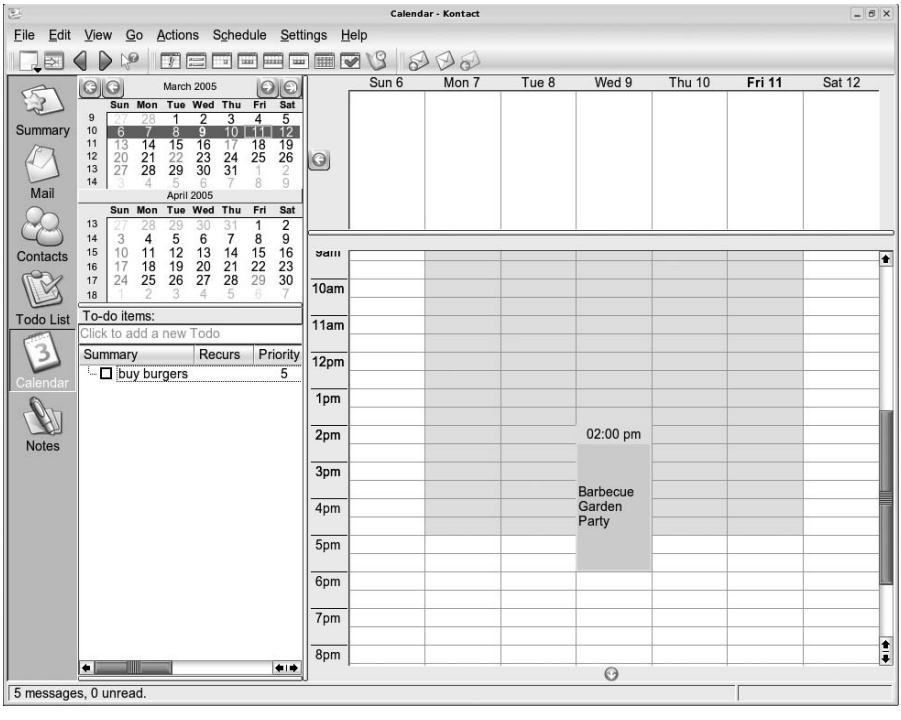


Рис. 8.46. *Создание нового события в Kontact*

легу как посетителя, а поскольку вечеринка без него будет не такой веселой, определить его присутствие как обязательное. По окончании ввода всей необходимой информации и закрытия диалога будет создано сообщение электронной почты и отослано на адрес вашего коллеги. Это сообщение будет состоять из текста с описанием события и дополнительной части, содержащей сведения о событии в определенном формате, который получил название *iTip* и описывается в RFC 2446.

На стороне получателя программа вашего коллеги для чтения почты Outlook распознает входящее сообщение как приглашение на событие и извлечет всю необходимую информацию из вложения. После этого у вашего коллеги будет запрошено, что делать с этим приглашением – принять, отвергнуть или пока принять, но под вопросом. Предположим, что ваш коллега не совсем уверен, что сможет прийти к вам, так как на эту среду он запланировал просмотр спортивного состязания, и потому принял приглашение, но пока не точно. После этого событие будет добавлено к его расписанию в Outlook, и вам будет отправлено сообщение, опять же содержащее специальное вложение iTip.

Как только сообщение вернется к вам, Kontact сообщит, что человек, которого вы пригласили, принял приглашение, но не точно, и запишет соответствующую информацию в ваше расписание. Когда ваш коллега решит более определенно отвергнуть или принять приглашение, будет послано новое сообщение, и статус контакта в вашем календаре обновится соответствующим образом. Если по каким-то причинам вы решите отменить встречу и удалите событие из своего рас-

писания, соответствующие уведомления автоматически будут разосланы всем заинтересованным лицам.

Описанные механизмы работают не только в случае событий, но и при назначении и рассылке задач другим людям, а также будут информировать вас о выполнении этих задач.

Чтобы добавить приглашенных в задачу из списка Todo List (Задачи) приложения Kontact, нужно щелкнуть правой кнопкой мыши на задаче, выбрать пункт контекстного меню Edit (Редактировать) и в открывшемся диалоге перейти на вкладку Attendees (Приглашенные). Разумеется, аналогичной функциональностью обладают все остальные клиенты, такие как Evolution или Mozilla, хотя сами диалоги несколько отличаются друг от друга.

Существует стандарт Интернета, описывающий формат, подобный формату iTip (или iCal, на котором основан iTip), обмена информацией о контактах, названный *vCard*. Например, чтобы сообщить свой новый адрес или номер телефона бабушке, которая для отслеживания своих контактов использует Mozilla для Windows, вы могли бы послать ей сообщение с вашей «визитной карточкой» в формате vCard (рис. 8.47). В Kontact для этого достаточно щелкнуть правой кнопкой мыши на записи в адресной книге и выбрать пункт контекстного меню Send Contact (Отправить контакт). Получившееся в результате сообщение электронной почты сможет быть воспринято большинством программ чтения электронной почты для Windows, Linux или Mac. Многие из этих программ дают пользователю возможность импортировать полученную «визитку» в свою собственную адресную книгу. На рис. 8.48 показано, как компонент электронной почты внутри оболочки Kontact отображает такое сообщение.

Как было продемонстрировано выше, взаимодействия между членами коллектива вполне можно организовать, используя лишь механизмы электронной почты. Такой порядок обладает следующими преимуществами: во-первых, для организации взаимодействий не требуется выделенный сервер, и, во-вторых, для успешного взаимодействия совершенно не важно, какими операционными системами и клиентами электронной почты пользуются члены коллектива. С дру-

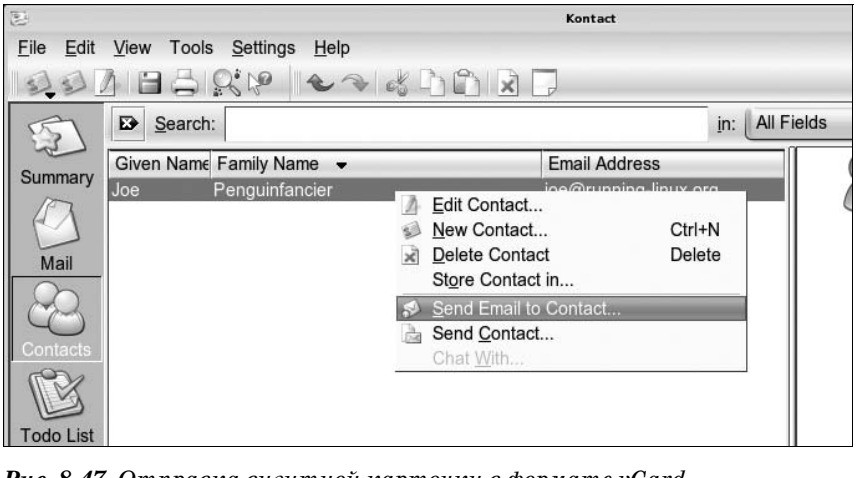


Рис. 8.47. *Отправка визитной карточки в формате vCard*

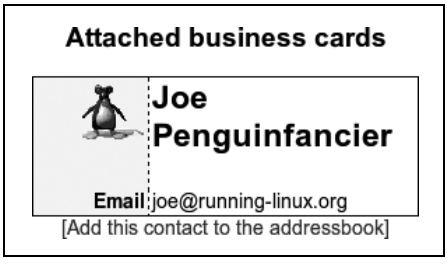


Рис. 8.48. *Принятая визитная карточка в формате vCard*

гой стороны, при использовании такой схемы будет очень сложно организовать ведение расписания, общего для нескольких человек, или предоставить доступ для чтения к централизованному хранилищу информации. В подобных ситуациях имеет смысл использовать сервер для рабочих групп.

Серверные решения для рабочих групп

Как платформа Linux поддерживается широким диапазоном серверных решений для рабочих групп, включая как проекты с открытыми исходными текстами, так и коммерческие продукты. Все они предлагают базовый набор функциональных возможностей для обмена электронной почтой, ведения расписаний и адресных книг, а также управления задачами, но помимо этого могут иметь различные расширения, с помощью которых можно реализовать управление ресурсами, следить за ходом выполнения задач и даже выполнять планирование проекта. Обычно такие системы допускают расширение своих функциональных возможностей с помощью дополнительных модулей, которые реализуют функции, отсутствующие в стандартном наборе. Такие модули иногда распространяются самим производителем, но нередко создаются сторонними фирмами или отдельными разработчиками.

В следующих ниже разделах описываются самые известные решения, распространявшиеся бесплатно на момент написания книги, с их назначением и особенностями.

Kolab

Проект Kolab вырос из контракта, заключенного Федеральным агентством безопасности Германии в области информационных технологий с группой компаний, на разработку серверного решения для рабочих групп, доступ к которому можно было бы осуществлять с помощью Microsoft Outlook и клиентов KDE для Linux. Разработчиками была создана серия концептуальных документов, описывающих реализацию сервера (получившего название Kolab 1 и Kolab 2). Они также предусмотрели возможность доступа к серверу и управление данными с помощью клиента Kontact для KDE. Дополнительно были разработаны модуль расширения с закрытыми исходными текстами для MS Outlook и веб-клиент.

Реализация сервера (Kolab 2) была выполнена на основе популярных свободно распространяемых программных компонентов, таких как сервер Cyrus IMAP для хранения электронной почты, агент транспортировки электронной почты

Postfix, служба каталогов OpenLDAP и веб-сервер Apache. Это полностью автономная система, которая устанавливается в операционную систему Linux и не имеет каких-либо внешних зависимостей. Уникальность сервера Kolab в том, что для хранения групповой информации он использует, в отличие от многих других решений, не реляционные базы данных, а почтовые папки сервера IMAP. Наконец, он реализует унифицированный интерфейс управления, написанный на языке программирования PHP.

Сервер Kolab предоставляет пользователям возможность совместного использования расписаний и папок контактов на основе механизма разграничения прав доступа для групп и отдельных пользователей. Кроме того, он предлагает возможность управления распределением списков и ресурсов, таких как комнаты или автомобили, и предоставляет возможность проверить занятость людей и ресурсов. В сервере предусмотрена также возможность делегирования прав, когда одни пользователи могут выполнять какие-либо действия от имени других пользователей, например распоряжения могут рассылаться секретарем от имени руководителя.

Дополнительные сведения о Kolab можно найти на сайте *http://www.kolab.org*.

OpenGroupware.org

Проект разработки сервера для рабочих групп (под названием OGo) возник, когда компания Skyrix Software AG открыла исходные тексты своего коммерческого продукта и продолжила работу над продуктом в составе сообщества уже в качестве основного вкладчика. Этот шаг положительно отразился на развитии как самой компании, так и проекта.

Сервер OGo предоставляет доступ к электронной почте, расписанию, контактам, документам и управлению задачами на основе веб-интерфейса. В дополнение к веб-интерфейсу доступ ко всем данным можно получить посредством различных стандартных протоколов, что сделало возможным использование таких клиентов, как Kontact или Evolution. Для пользователей Windows были разработаны модули расширения Outlook, правда, в виде коммерческого продукта. Пользователи могут совместно использовать календари, адресные книги, списки задач и создавать произвольные ассоциации между отдельными записями.

Для обеспечения полной функциональности сервера OGo необходимо выполнить установку некоторых дополнительных компонентов, таких как сервер IMAP, сервер баз данных PostgreSQL, агент транспортировки электронной почты и службы каталогов, например OpenLDAP.

Дополнительные сведения о серверном решении OGo можно найти на сайте *http://www.opengroupware.org*.

phpGroupWare и eGroupware

Решения phpGroupWare и eGroupware имеют общую программную основу, реализованную на языке программирования PHP, и предлагают функциональные возможности серверов для рабочих групп прежде всего через веб-интерфейс. Пользователи имеют возможность управлять своим расписанием, просматривать расписания и контактную информацию своих коллег, управлять файлами, заметками и сообщениями с новостями. Кроме того, в комплект входят несколько дополнительных приложений.

Оба сервера устанавливаются поверх существующего веб-сервера и сервера баз данных и могут использовать сервер электронной почты IMAP для организации обмена почтовыми сообщениями.

Дополнительные сведения о phpGroupware и eGroupware можно найти на сайтах *http://www.phpgroupware.org* и *http://www.egroupware.org*.

OPEN–XCHANGE

Сервер OPEN-XCHANGE начинался как коммерческий продукт, но впоследствии был передан под действие свободных лицензий. Подобно многим другим решениям он основывается и взаимодействует с другими серверными компонентами, такими как веб-сервер Apache и OpenLDAP. При установке поверх этих серверов данное решение предлагает несколько стандартных модулей, таких как календарь, контакты и задачи, предоставляет возможность управления документами и руководства проектом, реализует дискуссионный форум, базу знаний и содержит компоненты доступа к электронной почте через веб-интерфейс.

Технологически OPEN-XCHANGE отличается от многих других решений тем, что реализован на основе технологий Java. Это повышает его привлекательность в случае необходимости интеграции с другими Java-приложениями.

Дополнительные сведения об OPEN-XCHANGE можно найти на сайте *http://www.open-xchange.org*.

Продукты с закрытыми исходными текстами

Помимо свободно распространяемых решений с открытыми исходными текстами, описанных в предыдущих разделах, существует несколько коммерческих альтернатив. Все они обладают широкими возможностями и поддерживают в качестве платформы операционную систему Linux и некоторые другие операционные системы. Из наиболее известных можно назвать Novell Groupwise, Novell SUSE Linux Openexchange (основан на OPEN-XCHANGE), Lotus Notes & Domino, Oracle Groupware, а также Samsung Contact и Scalix (оба основаны на HP Openmail). Дополнительную информацию об этих продуктах можно найти на веб-сайтах их поставщиков и производителей.

LDAP: доступ к глобальным адресным книгам

Одно из преимуществ, которые дает организация централизованного хранения информации, состоит в том, что изменения и дополнения сведений производятся в одном месте, после чего все они немедленно становятся доступны всем пользователям. Это особенно важно для контактной информации, которая может часто изменяться и быстро устаревать. Возможность быстрого и гибкого поиска по большому числу контактов – это одно из требований, приобретающих особую важность в крупных организациях. Для удовлетворения этого требования была разработана так называемая *служба каталогов* вместе со стандартным протоколом, посредством которого производится доступ к информации и выполняются запросы. Этот протокол получил название *облегченный протокол доступа к каталогам* (LDAP – Lightweight Directory Access Protocol). Существует множество реализаций этого протокола, включая реализацию с открытыми исходными текстами OpenLDAP и Microsoft Active Directory (с характерными расширения-

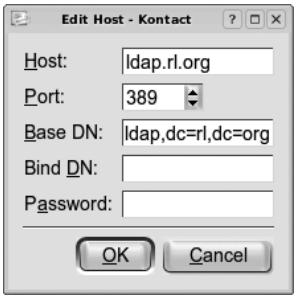


Рис. 8.49. *Добавление нового сервера LDAP в Kontact*

ми Microsoft). Реализация OpenLDAP может интегрироваться с большинством систем для рабочих групп, описанных в предыдущих разделах.

Компоненты адресной книги всех основных пакетов программного обеспечения для рабочих групп позволяют администратору связывать их с одним или несколькими серверами LDAP, после чего клиенты электронной почты могут отправлять запросы на получение контактной информации и использовать эту информацию в функции автозавершения адресов электронной почты при составлении электронных писем. Пример настройки запроса к серверу LDAP в Kontact приводится на рис. 8.49.

Здесь нужно указать сетевое имя сервера, который будет принимать запросы, номер порта (обычно используется значение по умолчанию) и так называемое *имя базового каталога*, которое определяет точку начала поиска в иерархии LDAP. Выбор имени базового каталога поможет приспособить запросы LDAP под нужды пользователей. Например, если компания имеет глобальную адресную книгу с подкаталогами для каждого из своих крупных отделений, тогда наверняка было бы предпочтительнее выполнять поиск только в локальном подкаталоге, а не по всей адресной книге. В этом случае администратор сервера должен сообщить пользователям, какие значения следует вводить в это поле. Если сервер принимает запросы только от зарегистрированных пользователей, необходимо будет ввести пароль доступа.

При наличии доступа к LDAP можно для примера пробовать начать составление нового электронного письма в Kontact и ввести чье-нибудь имя в строке с адресом получателя. Примерно через секунду появится список совпадений, найденных в центральной адресной книге на сервере LDAP. После этого можно просто выбрать из списка того адресата, который необходим в данном случае. Кроме того, все пакеты программного обеспечения для рабочих групп предоставляют возможность поиска и отображения контактной информации. В приложении Kontact диалог запроса можно вызвать, щелкнув на кнопке LDAP Lookup (Поиск адресов в каталоге LDAP), расположенной на панели инструментов, или выбрав одноименный пункт в меню Tools (Сервис).

Управление личными финансами

К настоящему моменту вы, возможно, уже заметили, что практически для любых видов деятельности существует программное обеспечение, распространяе-

мое с открытыми исходными текстами. Управление финансами – одно из самых распространенных занятий, таким образом, факт наличия приложения с открытыми исходными текстами для этого вида деятельности уже не должен вызывать у вас удивление. Это приложение называется *GnuCash*.

GnuCash – это ответ мира открытых исходных текстов на появление популярных коммерческих приложений управления личными финансами, таких как Microsoft Money и Intuit Quicken. Хотя в GnuCash нет «бантиков» и «рюшечек», присутствующих в этих приложениях, тем не менее в нем есть все, что необходимо для ведения личного или семейного бюджета. С помощью GnuCash можно следить за своими доходами и расходами, проверять сберегательные счета, следить за долгами, инвестициями и активами, такими как автомобили или недвижимость. Вы сможете проследить прошлые траты, чтобы разобраться с тем, куда шли ваши деньги, выяснить баланс, убедиться, что никаких неприятностей в ближайшем будущем не ожидается, а также прогнозировать свое финансовое благополучие на ближайшее и отдаленное будущее.

Если ранее вы уже пользовались такими приложениями, как Money или Quicken, то в GnuCash вы столкнетесь с некоторыми сюрпризами. По сравнению с упомянутыми приложениями интерфейс GnuCash выглядит достаточно аскетично. Здесь вы не найдете бесконечного числа диалогов настройки и мастеров и не сможете произвести электронные платежи. Вместо этого сразу после запуска GnuCash представит вам перечень счетов. Двойной щелчок на счете откроет его журнал (который выглядит в точности как страница чековой книжки). Здесь вы сможете провести какие-либо операции со счетом, и в результате баланс каждого счета будет отображен в списке счетов. Вы можете просмотреть различные отчеты и получить полное представление о состоянии своих финансовых дел. Это почти все, что есть в GnuCash.

Эта простота – достоинство, а не недостаток. Когда речь заходит о финансах, то чем проще, тем лучше. Еще одно важное отличие GnuCash от других приложений имеет отношение к способу, которым учитывается движение ваших денег. Более подробно об этом мы поговорим в разделе «Счет» далее в этой главе.

Введение

Запустите приложение GnuCash с помощью меню рабочего стола или выполнив команду *gnucash* из командной строки. На экране появится заставка, отображающая ход загрузки модулей. После того как приложение загрузится, заставка сменится окном диалога Tip of theDay (Совет дня) и Welcome to GnuCash! (Добро пожаловать!).

В диалоге Tip of the Day (Совет дня) всякий раз будет содержаться разная информация. В этом диалоге можно последовательно просматривать подсказки, щелкая на кнопках Prev (Назад) и Next (Далее). Рекомендуется оставить в настройках показ этого диалога при запуске, потому что информация, содержащаяся в нем, некоторое время может представлять для вас определенный интерес, однако, если вы предпочтете не видеть этот диалог, его запуск всегда можно будет отменить, сняв флажок Display this dialog next time (Показывать этот диалог в следующий раз). Окно можно закрыть, щелкнув на кнопке Close (Закрыть), но только после того, как будет дан ответ на вопрос.

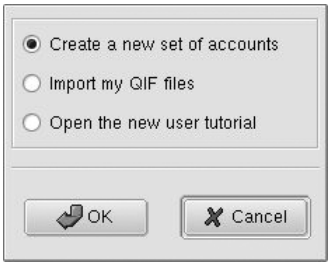


Рис. 8.50. *Диалог Welcome to GnuCash! (Добро пожаловать!) приложения GnuCash*

Диалог Welcome to GnuCash! (Добро пожаловать!) (рис. 8.50) демонстрируется только при первом запуске GnuCash. Это дает возможность создать новую группу счетов, импортировать данные из Quicken (файлы QIF) или запустить программу обучения. В данном примере мы создадим новый набор счетов, что является выбором по умолчанию, поэтому просто щелкните на кнопке OK.

После этого будет запущен *друид* New Account Hierarchy Setup (Создание новой иерархии счетов). Друид в Linux – это аналог мастера в Windows. Оба они представляют собой серию диалогов, которые проведут вас через необходимые операции настройки, выполняя за вас сложные задачи. Первое окно, которое вы увидите в друиде New Account Hierarchy Setup (Создание новой иерархии счетов), – это его описание. Щелкните на кнопке Next (Далее), чтобы приступить к процессу создания.

Выбор валюты

На рис. 8.51 приводится диалог выбора валюты. По умолчанию выбирается валюта в зависимости от региональных настроек, в данном случае это USD (US Dollar). Если вы используете другую валюту, выберите ее из раскрывающегося списка. После этого щелкните на кнопке Next (Далее).

Выбор категории набора счетов

На рис. 8.52 приводится перечень предустановленных типов счетов. Каждый из предлагаемых вариантов создает один или более счетов. Имеется возможность выбора нескольких вариантов (например, можно выбрать простую чековую книжку (A Simple Checkbook) или ссуду на приобретение автомобиля (Car Loan)), но пока выберите категорию A Simple Checkbook (Простая чековая книжка). Когда

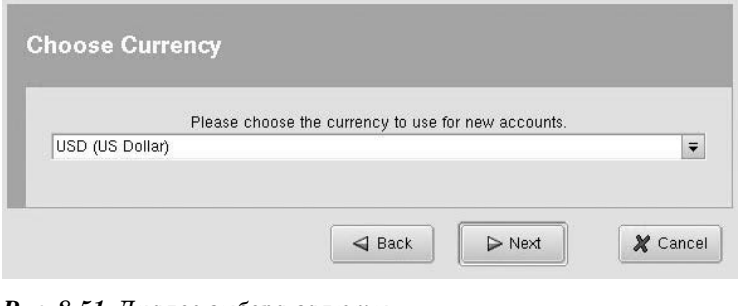


Рис. 8.51. *Диалог выбора валюты*

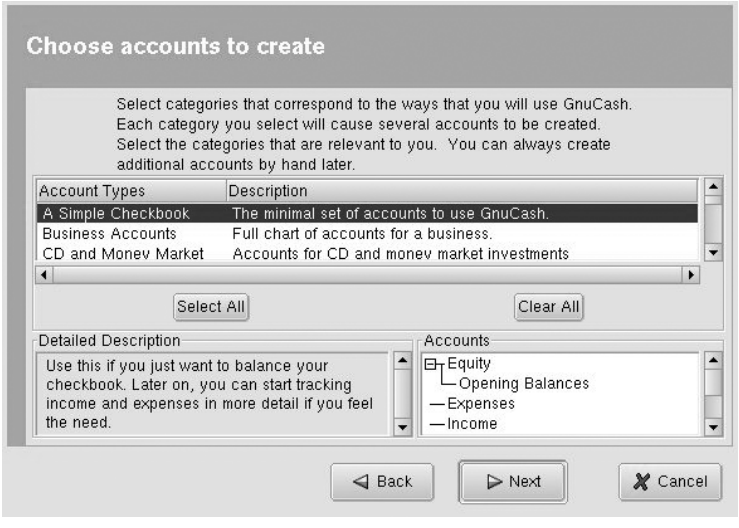


Рис. 8.52. *Страница создания счета*

тип выбран, внизу появляется его описание и список счетов, которые будут созданы. Не стоит беспокоиться по поводу количества доступных счетов. Пока это кажется странным, но к концу главы все прояснится. После этого щелкните на кнопке Next (Далее).

Ввод начального сальдо

Диалог, изображенный на рис. 8.53, дает возможность присвоить каждому счету начальное сальдо, то есть начальную сумму денег на этом счете. Если нужно

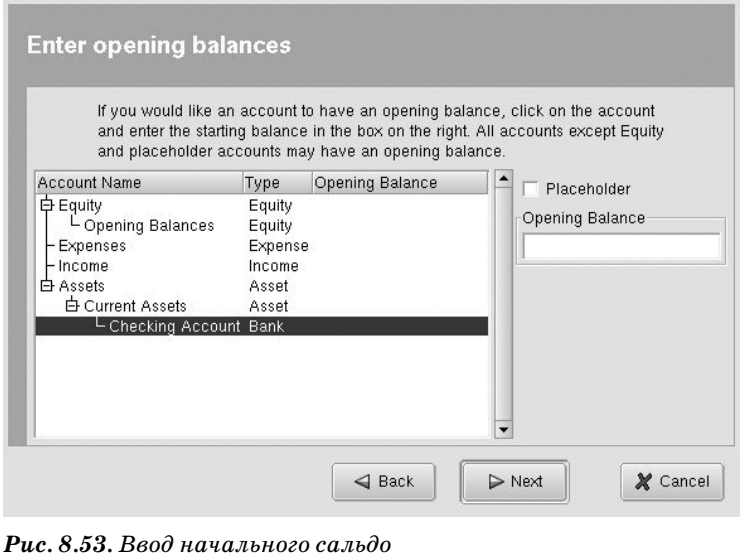


Рис. 8.53. *Ввод начального сальдо*

указать начальное сальдо текущего счета, просто щелкните на этом счете, чтобы выбрать его, и введите начальное сальдо. После этого щелкните на кнопке Next (Далее).

Завершение настройки счета

На этом настройку иерархии счетов в GnuCash можно считать законченной. Просто щелкните на кнопке Finish (Принять), и друид закроется.

Счет

Счет является одним из фундаментальных понятий в GnuCash. Счет – это место, куда деньги входят и откуда выходят. Когда большинство людей думают о счете, они представляют себе банковский счет. В GnuCash счетами является не только банковский счет, но и все остальное. Например, вы получаете на работе зарплату, откуда приходят деньги? Правильно, они приходят с вашего приходного счета. Вы потратили 30 долларов в магазине, куда ушли деньги? Они ушли на ваш счет «Питание».

Для отслеживания движения ваших денег в GnuCash используется метод двойного учета. Этот же метод используется профессиональными бухгалтерами и счетоводами для отслеживания движения миллиардов корпоративных долларов и правительственных активов, а теперь и вы будете использовать его (почувствовали себя важной персоной?). В методике двойного учета деньги всегда приходят с одного счета и поступают на другой счет. Всегда. Сумма на любом счете в каждый конкретный момент времени отражает количество денег, которое фактически находится на этом счете или прошло через него.

В GnuCash не все счета рассматриваются как равноценные. В этом введении мы рассмотрим только пять типов счетов: активы, долги, доходы, расходы и собственные средства.

Активы

Активы – это денежное выражение стоимости ваших вещей. Актив – это ваш текущий счет. Если деньги имеются на этом счете, значит, они у вас есть. Если у вас есть дом – это также актив. В GnuCash дом можно рассматривать как отдельный счет. Текущая стоимость дома – это сумма на счете. Вообще любой человек хотел бы, чтобы его активы постоянно росли.

Задолженность

Задолженность – это также денежное выражение стоимости ваших вещей. Единственное отличие – никто не хочет иметь задолженность! Если у вас есть дом, скорее всего, на него имеется закладная. Вы «владеете» обязательством выплатить определенную сумму своему кредитору. То есть сумма, которую осталось выплатить по долговым обязательствам, – это баланс данного счета. Балансы кредитной карточки, ссуды на приобретение автомобиля и долговые расписки – это примеры задолженностей. Любой человек хотел бы, чтобы его долги уменьшались.

Приход

В отличие от активов и долгов, приходный счет не представляет сумму, которая у вас имеется (по крайней мере, не напрямую). Приходный счет можно

представить себе как окно в чью-нибудь чековую книжку (как правило, вашего работодателя). Когда ваш работодатель выписывает вам чек, сумма чека записывается в расходную часть его книжки. Эту запись можно представить себе как запись в вашем приходном счете (представление той части его чековой книжки, которая касается вас), таким образом можно примерно понять суть приходного счета. Деньги, как правило, не остаются на этих счетах, они немедленно перетекают в один из ваших активов (обычно это ваша чековая книжка). Сумма на этом счете – это общая сумма, которую вам заплатили. Разумеется, никто не будет против, если эти счета будут увеличиваться.

Расход

Расходный счет также не представляет собой деньги, которыми вы владеете. Этот счет можно представить себе как депозитную (приходную) сторону чековой книжки того, кому вы выплачиваете деньги. Сумма на расходном счете – это общая сумма, которая на текущий момент была выплачена вами за товары или услуги. Сумму на этом счете уменьшить невозможно (разве что через скидки и возврат некоторых сумм), но любой хотел бы иметь возможность разумно управлять этим счетом.

Собственные средства (наличные)

Собственные средства – это самый малопонятный счет в данной группе. В принципе в финансовом мире существует формальное определение того, что является собственными средствами, но это далеко выходит за рамки данного введения. Проще всего этот вид счета можно представить как место, откуда взялся начальный капитал для всех остальных счетов. Выше уже говорилось, что в GnuCash деньги всегда приходят с некоторого счета и уходят на некоторый другой счет. А как быть с начальным сальдо – откуда оно взялось? Деньги не пришли с приходного счета, т. к. вы не получали зарплату для создания начального баланса. Таким образом, это ваши собственные средства.

Окно счетов в GnuCash

Окно со списком счетов (рис. 8.54) – это главное окно GnuCash. Здесь содержится список всех ваших счетов в текущем файле. Список счетов имеет древовидную структуру, потому что счета могут содержать вложенные счета (но об этом позже). А пока вам необходимо знать, что значок «плюс» слева от названия счета указывает, что данный счет является родительским по отношению к одному

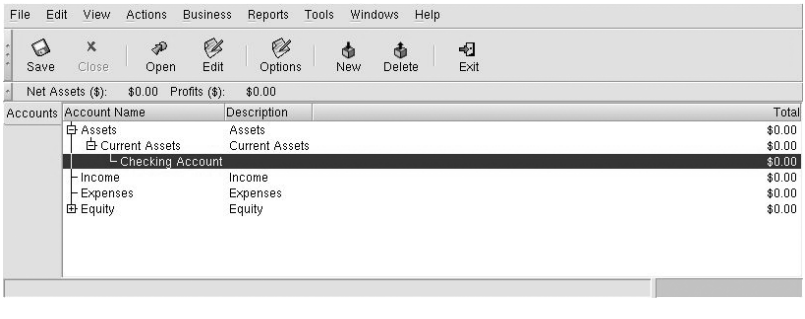


***Рис. 8.54.** Окно счетов GnuCash*

или более вложенным счетам, и что щелчок на этом значке раскрывает список вложенных счетов для этого счета. Таким образом, у вас есть возможность просмотреть этот список.

В списке счетов отображается название счета, его описание и общая сумма на этом счете. Если счет является родительским по отношению к одному или более субсчетам, общая сумма на нем складывается из сумм всех его субсчетов, а также из суммы самого родительского счета. Одиночный щелчок на счете делает его текущим. Щелчок правой кнопкой мыши вызывает контекстное меню с пунктами, позволяющими создать новый счет, удалить существующий, изменять параметры счета и выполнять множество других действий. Двойной щелчок на счете открывает соответствующую ему бухгалтерскую книгу или журнал. Более подробно о бухгалтерских книгах будет рассказываться далее.

Создание нового счета

Существует несколько способов создания нового счета. Самый простой заключается в том, чтобы щелкнуть правой кнопкой мыши на пустой области в окне со списком счетов. Другой путь – выбрать пункт главного меню File (Файл)→New Account (Создать счет). Сейчас мы создадим новый приходный счет, для этого просто щелкните правой кнопкой мыши в главном окне и в контекстном меню выберите пункт New Account (Создать счет).

На рис. 8.55 показано окно диалога New Account (Новый счет). Самое первое, что необходимо сделать, – это дать счету название. Поскольку в этот счет будут записываться все суммы, получаемые в качестве вознаграждения за труд, назовем счет Paycheck (Зарплата). Поля Account Code (Код счета) и Description (Описание) служат исключительно для личного пользования, поэтому в качестве кода счета можно указать, например, номер банковского счета, а в качестве описания – какую-нибудь содержательную информацию. Можно определить, в каком выражении будет исчисляться счет (поле Commodity (Товар)) так же, как это было сделано для основного файла со счетами. По умолчанию тип единицы измерения счета – валюта (USD, Euro, GBP и т. д.), и названия валюты берутся из основного файла, но вы можете изменить этот порядок и использовать другие виды валют (например, если вы шпион и имеете банковский счет в Цюрихе) или другие типы единиц измерения. Это бывает удобно при работе с акциями, облигациями и другими финансовыми инструментами. Доступные типы единиц измерения определяются выбранным типом счета.

Далее необходимо выбрать тип счета в поле Account Type. В этом поле вы найдете пять типов счетов, о которых говорилось выше, а также ряд других, используемых в специальных целях. Назначение нашего счета Paycheck (Зарплата) – учитывать доходы, поэтому найдите пункт Income (Приход) и выберите его. После того как будет выбран тип счета, необходимо установить значение поля Parent Account (Родительский счет). Счета могут вкладываться друг в друга, а это означает, что один счет может существовать как часть другого счета. У нас уже имеется счет с названием Income (Приход), поэтому щелкните мышью на значке «плюс» рядом с пунктом New Top Level Account (Новый счет верхнего уровня). В результате дерево счетов развернется, и перед вами появится список существующих счетов. Найдите в списке счет Income (Приход) и выберите его. Таким образом, счет Paycheck (Зарплата) станет вложенным счетом для счета Income (Приход).

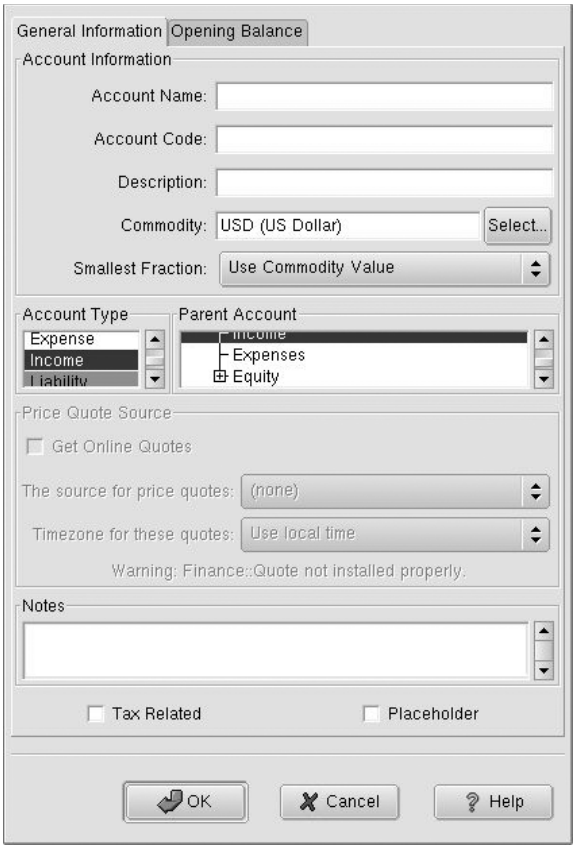


Рис. 8.55. *Диалог «Создать счет»*

Если поля Account Type (Тип счета) и Parent Account (Родительский счет) не видны, скорее всего, вам нужно просто изменить размеры окна, увеличив его высоту. Если окно по высоте равно высоте экрана, следовательно, вам необходимо изменить разрешение экрана. Эту операцию и в KDE, и в GNOME можно выполнить посредством диалогов.

Если бы этот счет предназначался для учета акций или других ценных бумаг (товаров), вы могли бы оперативно изменять ценовые пределы (чтобы, например, контролировать их стоимость). Однако описание подобных особенностей выходит за рамки данной главы. В поле Notes (Заметки) можно вводить свой пояснительный текст, который вы увидите позже, когда вновь вернетесь к этому экрану.

Наконец, в самом низу есть два флажка: Tax Related (Относящиеся к налогам) и Placeholder (Должностное лицо).[1] Флажок Tax Related (Относящиеся к налогам) связы-

[1] Термин «placeholder», который был переведен как «должностное лицо», точнее было бы перевести как «счет-контейнер», «счет-заполнитель», «счет для подстановки» или даже «группа счетов». – *Примеч. перев.*

вает этот счет с налоговой информацией, чтобы суммы некоторых налогов могли вычисляться автоматически. Использование этого параметра выходит за рамки нашего обсуждения.

Счет типа Placeholder (Должностное лицо) используется только для счетов, которые служат вместилищем других счетов. Например, у вас могут быть три источника дохода: работа, родители и частное предпринимательство по выходным. В этом случае вы могли бы поместить все три счета внутрь счета Income (Приход). Но сейчас счет Income (Приход) не должен иметь вложенных субсчетов, потому что у вас всего один источник доходов. Чтобы применить это правило, можно было бы установить флажок Placeholder (Должностное лицо) в диалоге настройки параметров счета Income (Приход). В результате журнал счета Income (Приход) оказался бы недоступным для внесения записей, и поэтому вам пришлось бы регистрировать свои доходы в одном из вложенных счетов. Определенно нет смысла использовать это правило для счета Paycheck (Зарплата), поэтому оставьте данный флажок сброшенным.

Щелкните на кнопке OK в диалоге New Account (Создать счет), и вы вернетесь к главному окну со списком счетов. Здесь вы сможете увидеть только что созданный счет Paycheck (Зарплата). Обратите внимание: этот счет был помещен внутрь счета Income (Приход), что нам и было нужно.

Чтобы изменить параметры существующего счета, нужно просто выбрать этот счет, затем правой кнопкой мыши вызвать контекстное меню и выбрать пункт Edit Account (Изменить параметры счета).

Удаление счета

Если счет был создан по ошибке, выберите его, щелкните на нем правой кнопкой мыши и выберите пункт контекстного меню Delete Account (Удалить счет), чтобы удалить счет из файла. Будьте осторожны, операция удаления счета затрагивает все записи, которые были сделаны для счета, что может нарушить баланс всех остальных счетов.

Не удаляйте счет только потому, что вы закрыли его (например, вы заплатили по кредиту и решили удалить информацию о нем или закрыли счет в старом банке). Даже при том, что счет закрыт, удаление счета приведет к потере информации о произведенных сделках и может нарушить баланс других счетов.

К сожалению, реального способа спрятать закрытые счета так, чтобы они больше не появлялись в окне счетов, не существует. Однако можно пойти на хитрость: создайте новый счет с флажком Placeholder (Должностное лицо) верхнего уровня с названием Closed (Закрытый) и переместите туда все закрытые счета (определив новый счет Closed (Закрытый) в качестве родительского). Поскольку родительский счет можно свернуть щелчком на значке «минус», все, что будет оставаться у вас перед глазами, – это лишь родительский счет. Эту хитрость нельзя назвать совершенной или особенно изящной, но она срабатывает.

Финансовые операции (транзакции)

Если счет составляет плоть GnuCash, то финансовые операции – ее кровь. Без наличия операций у вас будет всего лишь простой перечень счетов и ничего больше. В самом существовании такого перечня нет никакой пользы, и вы наверняка

захотите сделать что-то со всеми этими счетами. Запись сведений о проведенных операциях – это именно то, что позволяет GnuCash приносить пользу.

Операция в GnuCash – это запись, фиксирующая некоторый определенный случай. Этот случай обычно представляет собой передачу денег из одного места в другое, но это может быть также передача акций, долговых обязательств или недвижимого имущества. Если вам нужен конкретный пример, загляните в свою собственную чековую книжку. В ней вы найдете корешки чеков – это и есть записи об операциях. При использовании GnuCash вы просто ведете регистрационные записи об операциях в компьютере вместо своей чековой книжки (разумеется, благоразумный человек предпочтет сделать записи в обоих местах).

Ввод записи об операции

Чтобы ввести запись об операции, необходимо открыть окно журнала счета, как показано на рис. 8.56. Сделать это можно двойным щелчком на любом счете в списке. Давайте для начала создадим запись о такой операции, как приход. Предположим, что вы привели в порядок лужайку перед домом тетушки Алисы и она заплатила вам за ваш труд 25 долларов. А теперь посмотрим, как зафиксировать эту финансовую операцию в GnuCash.

Раскройте счет Assets (Активы), затем Current Assets (Оборотный капитал) и дважды щелкните на счете Checking Account (Текущий счет), чтобы открыть журнал.

Сегодняшняя дата уже указана в поле Date (Дата). Нажмите клавишу табуляции, чтобы перейти к следующему полю (нажатие клавиши Tab вызывает переход к следующему полю, а Shift+Tab – к предыдущему). В поле Num (Номер) можно ввести номер чека или любой другой номер, отождествляющий операцию. Представим себе, что тетушка Алиса выписала вам чек с номером 100, поэтому запишите число 100 в это поле.

С помощью клавиши табуляции перейдите к полю Description (Памятка) и введите что-нибудь содержательное. Вполне подойдет описание «Уход за лужайкой тетушки Алисы».

С помощью клавиши табуляции перейдите к полю Transfer (Передано). Это одно из самых важных полей. Помните, что в GnuCash деньги всегда идут с одного счета на другой. В данном случае деньги снимаются с вашего счета Income (Приход) и приходят на Checking Account (Текущий счет). Это поле (как и все остальные) обладает функцией автоматического дополнения. Просто введите In, и пе-

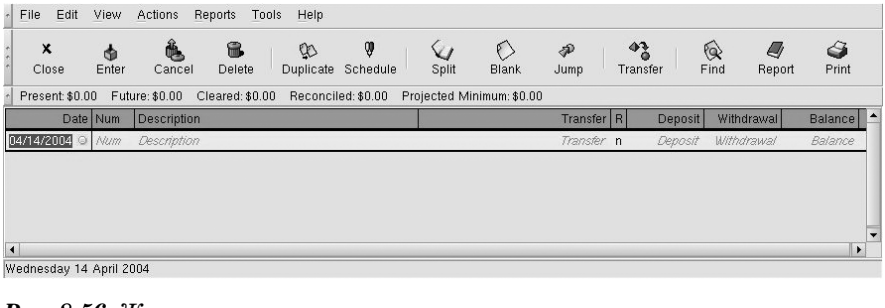


***Рис. 8.56.** Журнал счета*

ред вами должен появиться список счетов, в котором счет Income (Приход) будет выбран автоматически.

Поскольку вы получили эти деньги от тетушки Алисы, перейдите к полю Deposit (Депозит) и введите сумму 25.00.

После этого можно нажать клавишу Enter, и транзакция будет записана. Когда вы закроете окно журнала и взглянете на список свои счетов, то увидите, что суммы на счетах Income (Приход) и Checking Account (Текущий счет) увеличились на $25 каждый. Обратите внимание: родительский счет для Checking Account (Текущий счет) также показывает сумму $25. Родительский счет всегда отображает сумму всех вложенных в него счетов. Одного взгляда на окно программы достаточно, чтобы понять, что ваш приход составил $25 и $25 находятся на текущем счете.

Чтобы удалить запись об операции, нужно перейти в журнал с требуемой записью, щелкнуть правой кнопкой мыши на транзакции и выбрать пункт контекстного меню Delete (Удалить). В результате информация об операции будет удалена из всех счетов. Если говорить о денежной операции с тетушкой Алисой, упоминание о ней будет удалено из счетов Income (Приход) и Checking Account (Текущий счет).

Регистрация операций по частям

Предположим, что у вас на руках имеется расчетный листок о зарплате и вы готовы ввести информацию в GnuCash. Если вы не отличаетесь от подавляющего числа людей, тогда сумма, которую вам платят, не будет равна сумме, которую вы получаете. Разница идет в федеральные и местные налоги. Конечно, можно просто ввести сумму, которая определена к выдаче, но что, если вы захотите следить за доходами и расходами во всей их полноте, включая общий приход и уплату налогов? В GnuCash это можно сделать, описывая операции по частям.

Разделение операции на части позволяет описать несколько источников и получателей денег в одной операции. В данном случае в одной операции можно описать, что вами было заработано $500, из которых $100 пошли на уплату федерального налога, $50 – областного и еще $50 – местного, а $300 добавились к вашему текущему счету. Разделение операций на части позволяет сбалансировать приход из нескольких источников с расходом в нескольких направлениях. GnuCash позволяет вводить несбалансированные операции, разделенные на части, но постоянно будет напоминать об этом.

Чтобы ввести информацию об операции по частям, выполните следующие действия:

1. Откройте журнал счета. Разделенные операции обычно регистрируются внутри счета, соответствующего либо приходу, либо расходу. В случае зарплаты запись об операции обычно делается внутри текущего счета.
2. Введите дату и описание операции, как для любой другой.
3. Щелкните на кнопке Split (По частям), расположенной в панели инструментов.
4. Нажмите клавишу табуляции, чтобы перейти к первой части операции.
5. Опишите каждую из частей операции, как если бы это была обычная операция. Здесь есть один хитрый момент: поля Deposit (Депозит) и Withdrawal (Изъято) относятся к счетам, с которых приходят или на которые уходят деньги.

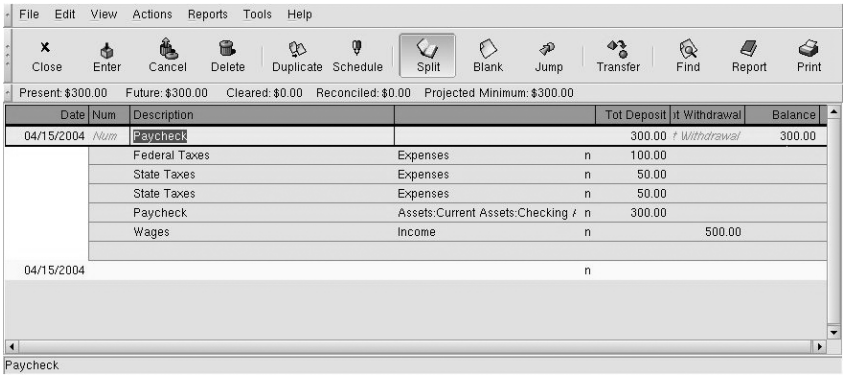


Рис. 8.57. *Пример операции, разделенной на части*

В данном случае $500 (заработная плата) изымается со счета Income:Paycheck (Приход:Зарплата) и поступает на несколько депозитов: Expenses (Расход) и Assets:Current Assets:Checking Account (Активы:Текущие активы:Текущий счет). Поначалу может показаться противоестественным, что налоги считаются депозитом, но, если вы перечитаете еще раз определение счета типа Expenses (Расход), все встанет на свои места. Воспользуйтесь рис. 8.57 как руководством по заполнению сведений об операции. Когда вы закончите с заполнением первой части, нажмите клавишу табуляции и переходите к заполнению следующей. На мой взгляд, это достаточно удобно – выделять приход денежных средств со счета Income (Приход) в виде отдельной части операции, как в данном примере. Это позволяет программе GnuCash автоматически подбивать баланс и упрощает последующие действия.

6. Для завершения работы с разделенной операцией нажмите клавишу Enter. Если операция получится несбалансированной (сумма прихода не совпадет с суммой расхода), GnuCash предупредит вас об этом и предложит варианты решения проблемы. Программа прекрасно «понимает», что ваши способности к математике могут быть не так хороши, как у компьютера, поэтому она будет выводить остаток неучтенных сумм в строке с последней частью операции. Как только баланс будет достигнут, запись о разделенной операции свернется в одну строку.
7. Чтобы увидеть состав разделенной операции, выберите ее левой кнопкой мыши и щелкните на кнопке Split (По частям), расположенной в панели инструментов.

Пример, приведенный на рис. 8.57, наглядно доказывает необходимость создания вложенных счетов для счета Expenses (Расход). Если бы заранее были созданы счета «Федеральный налог», «Областной налог» и «Местный налог», можно было бы сразу увидеть сумму каждого из налогов. Следуя этой методике, можно точно так же выделить различные категории расходов и доходов.

Запланированные операции

Вероятно, вам приходится оплачивать определенные счета каждый месяц приблизительно в одно и то же время, так что ввод сведений о таких операциях мо-

жет превратиться в надоедливую рутину. Однако программа GnuCash имеет в своем арсенале такое понятие, как запланированная операция, что позволяет регистрировать регулярные платежи автоматически через определенные интервалы времени. Чтобы запланировать операцию, выполните следующие действия:

1. В окне со списком счетов выберите пункт главного меню Actions (Действия)→ Scheduled Transactions (Запланированные транзакции)→Scheduled Transaction Editor (Редактор запланированных транзакций).
2. В появившемся диалоге щелкните на кнопке New (Создать).
3. В следующем диалоге введите название запланированной операции (например, «Оплата электроснабжения»), определите начальную дату, периодичность и конечную дату (если в этом есть необходимость).
4. В нижней части диалога находится шаблон описания операции. С его помощью можно указать сумму платежа за каждый интервал времени. Щелкните мышью на поле Description (Памятка) и создайте описание операции точно так же, как описание любой другой операции в текущем счете. Не забывайте, когда вы платите по счету, вы наверняка будете относить сумму платежа на расходный счет и забирать деньги с текущего счета. Шаблон должен отражать обе части операции. Данная операция будет производиться через указанные интервалы времени и отражаться на указанных в ней счетах.

Любую операцию можно быстро превратить в запланированную, щелкнув на ней правой кнопкой мыши и выбрав в контекстном меню пункт Schedule (Запланировать). Если вы испытываете затруднения с созданием шаблона операции вручную, можете попробовать схитрить, щелкнув на кнопке Advanced (Дополнительно) и посмотрев, как GnuCash автоматически создаст шаблон операции.

Отчеты

Как только вы потратите несколько месяцев на учет своей финансовой деятельности с помощью GnuCash, вы начнете понимать, какая магическая сила исходит от подробных отчетов о движении ваших денег. Одно дело иметь всю необходимую информацию, и совсем другое – организовать ее в том виде, который поможет вам определять тенденции или решать проблемы. К счастью, в GnuCash имеется широкий выбор отчетов, способных дать вам устойчивое понимание практически всех аспектов вашей финансовой жизни. В табл. 8.5 приводится перечень некоторых из самых общих отчетов и коротко описывается, какую информацию можно ожидать от них. Добраться до любого отчета можно с помощью меню Reports (Отчеты) в окне со списком счетов.

Для большинства из этих отчетов, чтобы они были действительно информативны, необходимо подробно структурировать дерево счетов. Например, если все свои расходы вы будете отправлять на один большой счет Expenses (Расходы), то тогда авансовый отчет (отчет о расходах) покажет, что 100% денег идут на счет Expenses (Расходы), что само по себе и так известно. Чтобы получить от отчетов максимум отдачи, необходимо структурировать дерево счетов так, чтобы каждая категория расходов имела свой счет внутри основного счета Expenses (Расход), то же самое относится к группам счетов Income (Приход), Liabiluty (Задолженность) и Assets (Активы). Чем более структурированным будет дерево счетов, тем больше информации удастся выжать из GnuCash.

Таблица 8.5. Отчеты GnuCash

Отчет	Что дает этот отчет
Итоги по счетам	Общее представление о балансе каждого из счетов
Гистограмма активов/диаграмма активов	Позволяет увидеть, как разделен ваш капитал. Для большинства людей их собственный капитал в первую очередь составляют дом, счет в банке и в пенсионном фонде
Гистограмма задолженностей/диаграмма задолженности	Представляет задолженности в процентном отношении. Для большинства людей самую значительную долю в общей задолженности составляют дом и автомобиль. Меньшую часть составляют кредитные карточки и ссуды на обучение и покупку товаров
Гистограмма собственного капитала	Активы – Задолженности = Собственный капитал. Данный отчет являет собой графическое представление этой формулы. Каждый человек стремится к тому, чтобы синие и зеленые части столбиков на диаграмме были как можно выше, а красные – как можно ниже
Гистограмма расхода/диаграмма расхода	Показывает, куда уходят ваши деньги. Если вы тратите 80% всех денег на приобретение одежды, данный отчет сообщит вам об этом (при условии, что структура счетов была выбрана вами правильно)
Гистограмма прихода/диаграмма прихода	Показывает, откуда берутся ваши деньги. Вы можете полагать, что большую часть денег приносит вам ваша работа, однако данный отчет способен удивить вас тем, сколько денег приходит к вам из других источников, например от родителей и работы по договорам (при условии, что структура счетов была выбрана вами правильно)

По умолчанию GnuCash создает отчеты на период времени от начала текущего года до текущей даты. Однако отчетный период можно изменить (например, получить отчет о расходах за апрель), достаточно лишь щелкнуть на кнопке Options (Настроить), расположенной на панели инструментов.

При вызове отчета с левой стороны главного окна создается панель вкладок, где будет находиться вкладка с названием отчета, выше нее вы должны увидеть вкладку Accounts (Счета). Используйте вкладки для перехода между окном со списком счетов и отчетом. Чтобы закрыть отчет, нужно щелкнуть на кнопке Close (Закрыть).

Щелчок на кнопке Exit (Выйти) закроет программу GnuCash, но отчет не исчезнет, он будет автоматически открыт при следующем запуске программы!

Примеры из реальной жизни

Изучение основ GnuCash по книге – это одно, а повседневное пользование этой программой – совсем другое. Вы уже видели, как ввести сведения о своей зарплате так, чтобы можно было учесть общую сумму начислений и налоговые отчисления. Ниже приводится еще несколько примеров из реальной жизни, чтобы дать вам возможность начать с решения самых распространенных задач.

Поход в гастроном

Ранее уже отмечалась важность настройки достаточно подробной структуры счетов, но еще не говорилось о том, как это проще сделать. Не обязательно заранее подготавливать структуру счетов. Все необходимые счета можно создавать по ходу ведения своих дел.

Вот как это делается:

1. Откройте журнал счета Checking Account (Текущий счет).
2. Создайте новую операцию с сегодняшней датой и описанием `Гастроном`.
3. В поле Transfer (Передано) введите символы `Ex`, в результате автоматически будет выбран счет Expenses (Расход). С помощью клавиши «стрелка вправо» довершите начатое функцией автодополнения. Теперь введите строку `:Продукты`. Символ двоеточия подсказывает программе GnuCash, что она должна создать внутри счета Expenses (Расход) вложенный счет с названием Продукты.
4. Нажмите клавишу Enter, чтобы закончить ввод новой категории, и затем клавишу табуляции, чтобы выйти из поля Transfer (Передано).
5. На экране появится диалог с запросом на создание нового счета Expenses:Продукты (Расход:Продукты). Щелкните на кнопке OK.
6. Появится диалог New Account (Создать счет) с уже установленными значениями по умолчанию, поэтому просто щелкните на кнопке OK.
7. Затем нужно перепрыгнуть через поле Deposit (Депозит), ввести в поле Withdrawal (Изъято) число `50.00` и нажать клавишу Enter.

Поздравляем! Вы не только описали свой поход в гастроном, но и создали дополнительный счет для этого. В будущем все расходы на продукты можно будет относить на этот счет, что даст возможность быстро узнать точную сумму расходов на питание при беглом просмотре содержимого списка счетов.

Функция автоматического дополнения очень удобна. Как только будут созданы вложенные счета, ввод символа двоеточия после имени счета откроет список счетов, вложенных в него.

Получение налогового возмещения

Большинство людей воспринимают налоговые возмещения как доход, но это не так: налоговые возмещения – это скидки (уступки). Если вы записываете суммы налогов с каждой зарплаты, запись о налоговых возмещениях – это просто операция возврата денег с расходного счета на ваш текущий счет.

Вот как это делается:

1. Откройте журнал счета Checking Account (Текущий счет).
2. Создайте новую операцию с сегодняшней датой и описанием `Налоговые возмещения`.
3. Поскольку уплату федеральных налогов мы описывали как переход денег со счета с зарплатой на счет Expenses (Расход), введите в поле Transfer (Передано) название счета `Expenses`.
4. Введите в поле Deposit (Депозит) число `50.00`.
5. Нажмите клавишу Enter, чтобы завершить операцию.

Теперь на вашем текущем счете появилось $50, но если вы взглянете на окно со списком счетов, то заметите, что приход не увеличился. Однако увеличилась общая сумма активов, а сумма счета Expenses (Расход) уменьшилась. Это в точности описывает происходящее, когда вы получаете налоговое возмещение. Теперь вы уже не будете дурачить себя мыслью, что каждый год зарабатываете дополнительные деньги. Вы их уже заработали, просто они вернулись к вам обратно!

Покупка автомобиля

Приобретение автомобиля всегда связано с большими расходами. А если вы приобретаете его в кредит, покупка становится еще дороже. К счастью, GnuCash может отследить каждый цент, потраченный на покупку, и определить, какая часть ежемесячной выплаты идет в счет погашения кредита, а какая – на уплату процентов по кредиту. Процесс описания покупки автомобиля – это отличный пример того, как следует описывать покупку дома или ссуду любого другого типа.

Описание покупки производится по следующему сценарию: вы только что приобрели совершенно новый автомобиль за 20 000 долларов. Первоначальный взнос составил $5000. Кредит будет погашаться в течение 60 месяцев, по $400 ежемесячно. Вероятно, кредитор выдал вам график погашения кредита, где описано, какая часть суммы идет в счет погашения кредита, а какая – на уплату процентов. Если вы не получили такой график, вам следует обратиться к своему кредитору или создать его самостоятельно, используя инструменты, доступные на вебсайте *http://www.bankrate.com*. Вы наверняка будете удивлены тем, сколько денег выплачивается в виде процентов. Чтобы оформить покупку автомобиля, необходимо:

1. Создать новый счет. Назовем его Кредит на автомобиль. В качестве родительского счета выберите пункт New Top Level Account (Новый счет верхнего уровня), а в качестве типа укажите Liability (Задолженность).
2. Создайте еще один счет с названием Автомобиль и поместите его внутрь счета Assets:Current Assets (Активы:Текущие активы), а в качестве типа укажите Assets (Активы).
3. Откройте журнал счета Автомобиль.
4. Создайте новую операцию. Введите описание операции: `Покупка автомобиля` в поле Description (Памятка) и затем щелкните на кнопке Split (По частям), расположенной на панели инструментов.
5. Первая часть операции будет описывать стоимость автомобиля. Введите а поле Description (Памятка) описание `Стоимость автомобиля`. В качестве счета укажите Assets:Current Assets:Автомобиль (Активы:Текущие активы:Автомобиль), а в поле Increase (Увеличено) укажите сумму $20 000.
6. Эти $20 000 должны откуда-то прийти. Самая первая часть – это ваш первоначальный взнос. Введите в поле Description (Памятка) описание `Первоначальный взнос` и укажите в качестве счета Assets:Current AssetsChecking Account (Активы:Текущие активы:Текущий счет), а в поле Decrease (Уменьшено) – сумму $5000. (Да, я понимаю, что в данном примере ваш банковский счет станет отрицательным. Не пытайтесь то же самое проделать дома!)
7. К сожалению, налоговые выплаты, оформление документов и техосмотр обойдутся вам еще в $1500. Введите описание `НДТ` (налоги, документы, техосмотр),

укажите счет Expenses (Расход) и в поле Increase (Увеличено) внесите сумму $1500.

8. Теперь общий баланс операции должен составлять $16 500 – это и будет сумма вашей задолженности. Введите описание Займ в поле Description (Памятка), в качестве счета выберите Кредит на автомобиль и в поле Decrease (Уменьшено) – сумму $16 500.
9. После завершения создания операции можно будет полюбоваться на плоды своего труда в окне со списком счетов.

Итак, вы владеете автомобилем уже целый месяц, и теперь пришло время платить по счетам. Быстрый взгляд на график погашения кредита показывает, что $300 пойдут в счет погашения кредита и $100 – на уплату процентов. Оформить эти платежи можно следующим образом:

1. Откройте журнал счета Checking Account (Текущий счет).
2. Создайте новую запись об операции. Укажите сегодняшнюю дату и введите в поле Description (Памятка) описание Платежи за автомобиль. Щелкните на кнопке Split (По частям), чтобы разделить операцию на части.
3. Общая сумма платежа составляет $400, поэтому введите в поле Description (Памятка) строку описания Платеж, укажите в качестве счета Assets:Current AssetsChecking Account (Активы:Текущие активы:Текущий счет) и запишите сумму $400 в поле Withdraw (Изъято).
4. Из общей суммы платежа $300 идут в счет погашения кредита, поэтому введите в поле Description (Памятка) описание Погашение кредита, в качестве счета укажите Кредит на автомобиль, а в поле Deposit (Депозит) запишите сумму $300.
5. Остальная часть выплат – это проценты по кредиту. Сумма баланса в $100 уже должна появиться в поле Deposit (Депозит), так что введите описание Проценты в поле Description (Памятка), в качестве счета выберите Expenses:Проценты (Расход:Проценты) (щелкните на кнопках Yes (Да) и OK после выхода из поля клавишей табуляции, чтобы создать вложенный счет). Нажмите клавишу Enter, чтобы завершить операцию.

Взгляните на окно со списком счетов. Здесь вы увидите, что сумма счета Кредит на автомобиль уменьшилась на $300, как и должно быть. Теперь вы не будете рассматривать платежи по кредиту за автомобиль как расходную часть бюджета – с его уменьшением будет уменьшаться и задолженность (увеличивая тем самым собственный капитал), и это будет происходить каждый месяц!

Предыдущая операция – превосходный кандидат на оформление в виде запланированной операции, которая должна производиться каждый месяц, избавляя вас от необходимости каждый раз вводить одни и те же данные. С каждой последующей выплатой обязательно сверяйте сумму выплаты по кредиту и процентам с графиком погашения кредита.

9

Мультимедиа

Эта глава рассказывает о мультимедиа в Linux. *Мультимедиа* – это довольно расплывчатый термин, которым злоупотребляют слишком часто. В данной главе под мультимедиа мы будем понимать нечто, что так или иначе связано с графикой, воспроизведением звука или видео.

Исторически средства мультимедиа были одним из наименее востребованных аспектов Linux как для пользователей, так и для разработчиков, не получавшим должного внимания в ее дистрибутивах, возможно, потому, что первоначально Linux рассматривалась большинством как операционная система для серверов. Лишь совсем недавно Linux всерьез стали рассматривать как настольное решение для среднего пользователя. Таким образом, чтобы добиться успеха в конкурентной борьбе за массового пользователя, поддержка мультимедиа стала насущной необходимостью.

За последние несколько лет положение дел с мультимедиа в Linux сильно изменилось. Теперь самые современные дистрибутивы автоматически обнаруживают и настраивают мультимедийные устройства, а также предоставляют типовой набор мультимедийных приложений. Несмотря на свое серверное прошлое, Linux прекрасно подходит для музыкальных и других мультимедийных приложений.

В первом разделе этой главы мы дадим краткий обзор некоторых мультимедийных понятий, таких как цифровой звук и видео, и опишем различные типы мультимедийных устройств. Читатели, знакомые с мультимедийными технологиями, могут пропустить этот раздел. Если вас действительно не интересует, как все это работает, или вы боитесь заблудиться в первых же предложениях этого раздела, не волнуйтесь – мультимедийными приложениями можно пользоваться, даже не понимая разницы между файлами форматов MP3 и WAV. В разделе «Фильмы и музыка: Totem и Rhythmbox» главы 3 описываются основные инструменты воспроизведения мультимедиа, входящие в состав большинства рабочих столов Linux.

Затем мы обсудим некоторые из проблем, связанных с поддержкой мультимедиа в ядре и являющихся предпосылкой к использованию аппаратных средств. Далее мы перейдем к рассмотрению приложений: сначала тех, что входят в состав некоторых наиболее популярных окружений рабочего стола, а затем более специализированных приложений, разбитых на различные категории. Для тех, кто пожелает заняться разработкой собственных приложений, мы кратко опишем

некоторые популярные инструменты и средства разработки. В заключение будет дан перечень печатных изданий и веб-ресурсов, где можно отыскать более подробную и актуальную информацию по данной теме.

Имейте в виду, мультимедиа – это область, где развитие Linux идет очень быстро. Новые технологии быстро превращаются из примитивных прототипов в средства массового использования. В 1996 году в книге о мультимедиа в Linux мы рассказывали о технологии под названием MPEG-1 уровня 3, или MP3. В то время эта технология была относительно малоизвестной и распространялась лишь некоторыми малозаметными веб-сайтами, а мой, на тот момент вполне современный, компьютер на базе процессора Intel 386 с тактовой частотой 40 МГц был едва в состоянии декодировать этот формат в режиме реального времени. Спустя всего несколько лет этот формат стал вездесущим и фактически превратился в стандартный формат файлов цифровой музыки в Интернете. В то же самое время другие технологии, которые выглядели многообещающими, оказались на обочине прогресса, причем зачастую совсем не по техническим причинам. Если у вас есть желание оставаться в курсе последних событий, обязательно ознакомьтесь с ресурсами, перечисленными в конце главы.

Между дистрибутивами Linux существуют некоторые незначительные различия. Большая часть сведений, которые приводятся в этой главе, справедлива для большинства дистрибутивов Linux, но для получения более подробной информации вам необходимо будет обращаться к документации, поставляемой с вашей системой, связываться с поставщиком дистрибутива и консультироваться с более опытными пользователями.

Основные понятия мультимедиа

В данном разделе мы очень кратко рассмотрим некоторые понятия, относящиеся к цифровому звуку, видео и звуковым картам. Понимание этих основ поможет разобраться в информации, которая последует далее в этой главе.

Преобразование сигналов в цифровую форму

Звук возникает при распространении волн переменного давления в среде, которой обычно является воздух. Это явление имеет *аналоговую* природу, что означает непрерывное изменение давления воздуха в некотором диапазоне значений.

Современные компьютеры являются *цифровыми*, то есть они работают с дискретными величинами, по существу, с двоичными единицами и нулями, обрабатываемыми центральным процессором компьютера. Для того чтобы компьютер мог обрабатывать звук, он должен преобразовывать аналоговую информацию звука в цифровой вид.

Аппаратное устройство, которое называется *аналогово-цифровым преобразователем*, преобразует аналоговые сигналы, такие как непрерывно изменяющиеся электрические сигналы микрофона, в цифровой формат, чтобы компьютер мог его обрабатывать. Аналогично *цифро-аналоговый преобразователь* преобразует цифровые значения в аналоговый вид, в котором они могут быть отправлены на аналоговое выходное устройство, например громкоговоритель. Звуковые карты обычно содержат несколько аналогово-цифровых и цифро-аналоговых преобразователей.

Процедура преобразования аналоговых сигналов в цифровую форму заключается в дискретизации, то есть взятии замеров через равные промежутки времени и сохранении этих значений в виде чисел. Процедура аналогово-цифрового преобразования не является совершенной, что приводит к некоторым потерям или искажениям информации. Два важных фактора, влияющих на качество представления аналоговых сигналов в цифровой форме, – это *точность* и *частота дискретизации*.

Точность представления – это диапазон чисел, представляющих значения отсчетов сигнала, обычно выражаемый количеством двоичных разрядов. Например, 8-разрядные отсчеты преобразуют величины аналогового звукового сигнала в одно из 2^8, или 256, дискретных значений. 16-разрядное квантование представляет звуковой сигнал с помощью 2^{16}, или 65 535, различных значений. Чем больше разрядов используется, тем точнее представляется сигнал и меньше ошибка квантования, возникающая при представлении сигнала дискретными значениями. Обратной стороной увеличения разрядности являются повышенная потребность в памяти и сложность аппаратуры, ведущая к ее удорожанию.

Частота дискретизации определяет интервал времени между последовательными измерениями аналогового сигнала. Правильно выражать ее количеством отсчетов в секунду, хотя иногда неформально и менее строго ее выражают в герцах. Чем ниже частота дискретизации, тем больше теряется информации об исходном аналоговом сигнале, и наоборот, при увеличении частоты дискретизации сигнал представляется точнее. *Теорема о частоте дискретизации*[1] утверждает, что для точного представления аналогового сигнала его отсчеты должны сниматься с частотой как минимум вдвое большей, чем самая высокая частота, присутствующая в исходном сигнале.

Человеческое ухо в идеальном случае различает звуки в диапазоне примерно от 20 до 20 000 Гц, поэтому для точного представления звука, слышимого человеком, нужна частота выборки, вдвое большая 20 000 Гц. В технологии записи CD используется частота 44 100 отсчетов в секунду, что согласуется с этим простым расчетом. Спектр человеческой речи в основном укладывается в 4000 Гц. В системах цифровой телефонии обычно используется частота 8000 отсчетов в секунду, что отлично подходит для передачи речи. Увеличение частоты дискретизации ведет к росту потребности в памяти и усложнению аппаратуры.[2]

При записи звука в цифровом виде возникают также проблемы количества каналов и формата кодирования отсчетов. Для стереозвучания нужны два канала. В некоторых аудиосистемах применяются четыре или более каналов.

Часто нужно соединять звуки вместе или изменять их громкость. Эта процедура микширования может осуществляться как в аналоговом виде (например, регулятором громкости), так и в цифровом виде (компьютером). Теоретически два оцифрованных сигнала можно микшировать вместе, просто складывая отсчеты, а изменять громкость можно, умножая значения отсчетов на некоторую константу.

---

[1] За рубежом известная как «теорема Найквиста», а русскоязычному читателю, возможно, более известная как «теорема Котельникова». – *Примеч. науч. ред.*

[2] Дальнейшее увеличение частоты дискретизации до значений, заметно превосходящих значения, обсуждаемые автором (частота Котельникова), не ведет к сколько-нибудь заметному росту точности воспроизведения. – *Примеч. науч. ред.*

До сих пор мы говорили о том, как записывать звук в виде цифровых отсчетов. Часто используются и другие технологии. *ЧМ-синтез* является старой технологией создания звука с помощью аппаратуры, которая обрабатывает сигналы различной формы, например волны синусоидальной или треугольной формы. Соответствующая аппаратура весьма проста и часто использовалась в звуковых картах первого поколения для синтеза музыки. Многие звуковые карты до сих пор поддерживают ЧМ-синтез для достижения обратной совместимости. В некоторых более новых картах используется технология синтеза на основе волновых таблиц, в которой ЧМ-синтез усовершенствован благодаря генерации звука с помощью цифровых образцов, хранящихся в самой звуковой карте.

MIDI служит сокращением от Musical Instrument Digital Interface (цифровой интерфейс музыкальных инструментов). Это стандартный протокол связи между электронными музыкальными инструментами. В число типичных устройств MIDI входят музыкальные клавиатуры, синтезаторы и ударные установки. MIDI оперирует не хранящимися образцами звука, а такими событиями, как нажатия клавиш на музыкальной клавиатуре. События MIDI можно записывать в MIDI-файл, благодаря чему музыкальное произведение представляется в очень компактном виде. MIDI наиболее популярен среди профессиональных музыкантов, хотя многие потребительские звуковые карты поддерживают интерфейс шины MIDI.

Форматы файлов

Мы поговорили о цифровом звуке, который обычно извлекается звуковой картой и хранится в памяти компьютера. Чтобы сохранить звук надолго, его необходимо представить в виде файла. Для этого существуют разные методы.

Самый простой метод состоит в том, чтобы записывать звук прямо в файл в виде простой последовательности байтов. Такие файлы часто называют *аудиофайлами RAW-формата*. Сами отсчеты сигнала могут кодироваться в различных форматах. Мы уже упоминали самые распространенные 8- и 16-разрядные форматы. Для заданной разрядности отсчеты могут кодироваться с помощью чисел со знаком или без знака, а когда отсчет занимает больше одного байта, необходимо указать принятый порядок следования байтов. Эти проблемы важны для передачи цифрового звука между программами или компьютерами, чтобы гарантировать использование ими одинакового формата.

Проблема с аудиофайлами формата RAW состоит в том, что сам файл не определяет ни точность, ни частоту дискретизации, ни представление данных. Эта информация совершенно необходима, чтобы интерпретировать файл правильно. Такие форматы представления цифрового звука, как WAV, добавляют в файл всю необходимую информацию в форме заголовка, чтобы приложения могли определить, как интерпретировать данные из файла. Эти форматы стандартизируют представление звуковой информации, позволяя перемещать ее между различными компьютерами и операционными системами.

Возможность хранения аудиосигналов в виде файлов упрощает работу со звуком, но имеет и недостаток: аудиофайлы имеют тенденцию быстро увеличиваться в размерах. Мы уже говорили о звуковых компакт-дисках, в которых используется 16-разрядное квантование уровня и частота дискретизации 44 100 отсчетов в секунду при двух каналах (стерео). Файл, вмещающий один час звучания

в формате CDDA (Compact Disc Digital Audio – формат компакт-дисков со звуковым содержимым), будет иметь размер более 600 Мбайт. Чтобы уменьшить объем данных, необходимых для хранения звука, были разработаны различные схемы компрессии. Один из подходов заключается в том, чтобы просто сжимать данные с помощью тех же алгоритмов, которые используются для компьютерных данных. Однако, если учесть особенности человеческого слуха, можно добиться более эффективного сжатия звуковых данных путем удаления тех составляющих звука, которые ухо не слышит. Такая обработка называется *сжатием с потерями*, поскольку сопровождается потерей информации. Однако при правильной реализации объем данных сокращается очень значительно, а потеря качества звука незаметна. Такой подход использует алгоритм сжатия звука MPEG-1 уровня 3 (MP3), который может обеспечивать коэффициент сжатия 10:1 относительно первоначального цифрового звука. Другим алгоритмом с потерями, позволяющим добиться аналогичных результатов, является Ogg Vorbis. Он популярен среди многих пользователей Linux, поскольку позволяет избежать проблем с патентными правами, связанных с кодированием MP3. Существуют алгоритмы компрессии, оптимизированные для человеческой речи, например кодировка GSM, используемая в некоторых цифровых телефонных системах. Алгоритмы, с помощью которых осуществляется кодирование и декодирование звука, иногда называются *кодеками*. Одни кодеки основаны на открытых стандартах, таких как Ogg и MP3, которые могут быть реализованы на основе опубликованных спецификаций. Другие, форматы которых хранятся в тайне, являются собственностью разработчиков или лиц, получивших патент на эту технологию. Примерами закрытых кодеков могут служить RealAudio компании Real Network, WMA корпорации Microsoft и QuickTime компании Apple.

До сего момента основное внимание мы уделяли звуку, теперь поговорим о видео. Проблема хранения изображения имеет много общего с проблемой хранения звука. В случае с изображениями отсчеты – это пикселы (элементы изображения), представляющие цвет, а точность оттенков цвета определяется количеством используемых битов. Чем большим числом битов описывается цвет, тем более точно могут быть представлены различные оттенки, но опять же за счет большего количества памяти. Обычно глубина цвета определяется 8, 16, 24 или 32 битами. Файл растровой графики просто хранит пикселы изображения в некотором предопределенном формате. Как и в случае со звуком, есть RAW-форматы файлов с изображениями, а есть форматы, в которых файлы содержат дополнительную информацию, позволяющую определить их тип.

Для сжатия файлов с изображениями используются различные методики. Могут использоваться стандартные алгоритмы сжатия, такие как *zip* и *gzip*. Для изображений, содержащих однотонные области, например линии, прекрасно подходит способ группового кодирования, при котором цвет задается сразу для группы точек растра. Как и в случае с аудиоформатами, существуют схемы сжатия с потерями, например JPEG, оптимизированные для хранения таких изображений, как фотографии, и специально спроектированные с целью обеспечить высокую степень сжатия при незначительных искажениях.

Чтобы распространить вышесказанное на видео, можно просто представить себе последовательность изображений, упорядоченных по времени. Совершенно очевидно, что такой способ представления видео должен порождать файлы гигантских размеров. В схемах сжатия, которые используются для хранения фильмов

DVD, применяются очень сложные алгоритмы, позволяющие хранить полные изображения как математическую разность между соседними кадрами, что дает возможность обновлять изображение при воспроизведении. Все это алгоритмы кодирования с потерями. Кроме того, фильмы также содержат одну или более звуковых дорожек и другую информацию, как, например, субтитры.

Выше уже упоминался формат CDDA, позволяющий сохранять приблизительно 600 Мбайт звука на одном диске. Для хранения обычных компьютерных данных на CD-ROM используется тот же самый физический формат, основанный на файловой системе, известный как формат ISO 9660. Он позволяет создавать обычные структуры каталогов, как в MS-DOS. Для расширения возможностей ISO 9660 было разработано расширение Rock Ridge, позволившее хранить файлы с длинными именами и набором дополнительных признаков, что сделало этот формат пригодным для использования в UNIX-подобных системах. Аналогичные функции выполняет файловая система Joliet, разработанная в Microsoft и используемая различными версиями Windows. CD-ROM может быть отформатирован как с расширением Rock Ridge, так и с расширением Joliet, что делает его читаемым и в UNIX-совместимых, и в Windows-совместимых системах.

CD-ROM изготавливаются на дорогостоящем промышленном оборудовании. CD-R (однократно записываемый компакт-диск) позволяет выполнить запись данных на относительно недорогом устройстве и может читаться на стандартном приводе для компакт-дисков. CD-RW (перезаписываемый компакт-диск) позволяет многократно записывать и стирать информацию.

Устройства DVD-ROM позволяют хранить на одном диске порядка 4,7 Гбайт данных, для чего используется тот же самый формат, что и для фильмов DVD. При наличии соответствующей аппаратуры и программного обеспечения декодирования персональный компьютер, оснащенный приводом DVD, может использоваться для просмотра фильмов DVD. Не так давно появились двухслойные диски DVD, которые обладают удвоенной емкостью.

По аналогии с CD-R диски DVD могут быть записываемыми, но в двух различных форматах – DVD-R и DVD+R. Во время написания этих строк оба формата пользовались примерно одинаковой популярностью, и некоторые комбинированные приводы поддерживали их оба. Точно так же и для перезаписываемых дисков DVD были разработаны два разных формата – DVD-RW и DVD+RW. Наконец, появился формат DVD-RAM, допускающий возможность произвольного доступа к данным для чтения-записи, как обычный жесткий диск.

Диски DVD-ROM могут быть отформатированы в файловую систему ISO 9660 с возможными расширениями Rock Ridge и Joliet. Однако достаточно часто эта задача выполняется с помощью файловой системы UDF (Universal Disc Format – универсальный дисковый формат), которая используется для записи видеофильмов DVD и прекрасно подходит для носителей данных большой емкости.

Для приложений, в которых нужно передавать звук через Интернет, иногда широковещательно для множества пользователей, звуковые файлы непригодны. Системы, способные передавать звук или иную мультимедийную информацию и воспроизводить ее в реальном времени, обозначают термином *«потоковое мультимедиа»* (*streaming media*).

Устройства мультимедиа

Теперь, когда мы обсудили концепции, лежащие в основе цифрового звука, посмотрим на используемую для обработки звука аппаратуру. Звуковые карты развивались аналогично другим картам периферийных устройств для PC. Карты первого поколения были рассчитаны на шину ISA и в большинстве своем обеспечивали совместимость с серией SoundBlaster производства Creative Labs. Появившийся стандарт ISA Plug and Play (PnP) был принят многими звуковыми картами, что упростило настройку, устранив необходимость в аппаратных перемычках. В современных звуковых картах обычно используется шина PCI, будь это отдельные карты или поддержка звука, интегрированная на материнской плате, но доступ к которой все равно осуществляется через шину PCI. Сейчас появился ряд звуковых устройств USB, наиболее популярными из которых являются акустические системы, управляемые через шину USB.

Некоторые звуковые карты теперь поддерживают развитые функции, такие как *круговое стереозвучание* через шесть каналов и цифровой ввод-вывод для подключения к системам домашнего кинотеатра. Обсуждение таких устройств выходит за рамки нашей книги, и соответствующих звуковых карт мы касаться не будем.

По аналогии со звуком в царстве видео существуют видеокарты, многие из которых совмещают в себе функции ускорителя трехмерной графики, обладают большими объемами встроенной памяти и иногда имеют более одного видеовыхода.

ТВ-тюнеры могут декодировать телевизионный сигнал и выводить его на монитор, нередко через видеокарту, что позволяет смешивать телевизионное изображение с компьютерным. Карты видеозахвата позволяют записывать видеосигнал на жесткий диск в режиме реального времени.

Мышь и клавиатура давно стали самыми обычными устройствами ввода данных, но кроме них Linux поддерживает разнообразные сенсорные экраны, планшетные дигитайзеры и джойстики.

Linux поддерживает большинство сканеров. Старые модели обычно подключались к компьютеру через параллельный порт или интерфейс SCSI. Некоторые из них выполняют обмен данными с использованием закрытых протоколов и по этой причине не поддерживаются в Linux. Более новые сканеры подключаются через интерфейс USB, хотя некоторые высокопроизводительные профессиональные модели вместо него используют FireWire (термин Apple, обозначающий стандарт, известный также как IEEE 1394), обладающий большей пропускной способностью.[1]

В последнее время значительно улучшилась поддержка цифровых камер благодаря появлению большего числа драйверов и благодаря тому, что производители камер начинают отдавать предпочтение более стандартизированным протоколам. Старые модели обычно подключались через последовательный порт или интерфейс SCSI. Более современные устройства используют интерфейс USB, если они вообще предоставляют возможность прямого кабельного соединения. Кроме

[1] Эта разница в пропускной способности ушла в прошлое с появлением стандарта интерфейса USB 2, которым укомплектованы практически все компьютеры последних лет. – *Примеч. науч. ред.*

того, многие из них выполняют запись изображения на стандартные модули флэш-памяти, которые могут выниматься из устройства и считываться на компьютере с помощью специального адаптера, подключаемого к компьютеру через порт USB или PCMCIA. С принятием стандартного протокола обмена данными с запоминающими устройствами USB большой емкости все устройства, совместимые с этим стандартом, должны поддерживаться операционной системой Linux. Ядро Linux представляет устройства USB большой емкости так, как если бы они были устройствами SCSI.

Ядро и проблемы драйверов

Тема настройки и сборки ядра будет обсуждаться в другой главе этой книги. А здесь мы рассмотрим лишь некоторые моменты, имеющие отношение к мультимедиа. Как уже говорилось выше, большинство карт мультимедиа подключаются к шине PCI и должны автоматически обнаруживаться и настраиваться ядром Linux.

Драйверы звуковых устройств

История развития драйверов звуковых устройств в Linux заслуживает того, чтобы быть упомянутой, потому что она поможет объяснить существующее ныне разнообразие. На начальной стадии разработки Linux (то есть до выхода версии ядра 1.0) Ханну Саволайнен (Hannu Savolainen) реализовал драйверы поддержки звука на уровне ядра для ряда популярных звуковых карт. Другие разработчики также сделали свой вклад, добавив в этот код поддержку новых функций и дополнительных карт. Эти драйверы, входящие в стандартную версию ядра, иногда называют OSS/Free – бесплатной версией Open Sound System.

Позднее Ханну поступил на работу в 4Front Technologies, компанию, занимающуюся разработкой коммерческих драйверов поддержки звука для Linux и ряда других UNIX-совместимых операционных систем. Эти улучшенные драйверы распространяются на коммерческой основе под названием OSS/4Front.

В 1998 году был организован проект Advanced Linux Sound Architecture, или ALSA, задачей которого стало создание новых звуковых драйверов для Linux и решение проблемы с отсутствием активной поддержки звуковых драйверов OSS. С учетом оценки прошлых ошибок и требований новой технологии звуковых карт возникла потребность в новых разработках.

Некоторые изготовители звуковых карт также создали драйверы для своих звуковых карт под Linux, в частности, это относится к серии Creative Labs Sound Blaster Live!.

В результате теперь предоставляется четыре разных комплекта звуковых драйверов для ядра. Это создает трудности при выборе необходимого звукового драйвера. В табл. 9.1 приведены преимущества и недостатки различных драйверов, что может помочь при принятии решения. Нужно также учесть, что в конкретном дистрибутиве Linux драйвер будет, скорее всего, один, и для выбора какого-либо другого потребуются дополнительные усилия.

Помимо драйверов, перечисленных в табл. 9.1, существуют исправления ядра, решающие проблемы с некоторыми звуковыми картами.

Таблица 9.1. Сравнительная характеристика драйверов звуковых карт

Драйвер	Преимущества	Недостатки
OSS/Free	Бесплатный Есть исходный программный код Входит в стандартное ядро Поддерживает большинство звуковых карт	Поддерживает не все звуковые карты Большинство звуковых карт не определяются автоматически Не поддерживает некоторые новейшие карты Отсутствует централизованное сопровождение
OSS/ 4Front	Поддерживает большое число звуковых карт Автоматически определяет большинство карт Имеется коммерческая поддержка Совместимость с OSS	Требуется оплата Исходные тексты закрыты
ALSA	Бесплатный Есть исходный программный код Поддерживает большинство звуковых карт Активно развивается и поддерживается Большая часть звуковых карт определяется автоматически	Не все звуковые карты поддерживаются Не полностью совместим с OSS
Коммерческие	Могут поддерживать карты, не поддерживаемые другими драйверами Могут поддерживать специфические функции аппаратного устройства	Исходные тексты могут быть недоступны Может отсутствовать официальная поддержка в ядре

Большинство звуковых карт поддерживаются в Linux тем или иным драйвером. Отсутствие поддержки наиболее вероятно для очень новых карт, для которых еще не разработаны драйверы, и для некоторых высококачественных профессиональных карт, редко используемых обычными потребителями. Достаточно актуальный список поддерживаемых карт можно найти в постоянно обновляемом документе «Linux Sound HOWTO», но часто лучше всего заняться поиском в Интернете и поэкспериментировать с драйверами, которые могут подойти к вашей аппаратуре.

Многие приложения для работы со звуком непосредственно обращаются к звуковым драйверам ядра, однако это служит источником проблемы: к звуковым устройствам ядра может одновременно обращаться только одно приложение. В графической настольной среде пользователь может пожелать одновременно воспроизводить файл MP3, сопровождать звуками операции оконного менеджера, получать звуковой сигнал при поступлении новой почты и т. д. Для этого требуется совместный доступ к звуковым устройствам со стороны различных приложений. Чтобы решить эту проблему, в современные графические среды Linux включается сервер звука, который берет на себя монопольное управление звуковыми устройствами и принимает запросы от приложений на воспроизведение звуков,

микшируя их вместе. Возможно также перенаправление звука на другой компьютер подобно тому, как X Window System позволяет отображать графические результаты работы программы на экране компьютера, отличном от того, на котором она выполняется. Настольная среда KDE использует сервер звука *artsd*, а в GNOME эту же задачу решает *esd*. Поскольку серверы звука являются относительным нововведением, пока еще не все приложения, работающие со звуком, их поддерживают. Нередко подобные проблемы удается разрешить с помощью программы-обертки, такой как *artswrapper*, которая переадресовывает обращения к звуковому устройству на вход сервера звука.

Установка и настройка

В этом разделе мы расскажем о том, как установить и настроить звуковую карту в Linux.

Трудоемкость этой процедуры различна в зависимости от конкретного дистрибутива Linux. По мере роста зрелости Linux некоторые дистрибутивы стали предоставлять возможность автоматического определения и настройки звуковых карт. Времена ручной установки перемычек на карте и устранения аппаратных конфликтов уходят в прошлое в связи с повсеместным использованием шины PCI. Если вам повезло и в вашем дистрибутиве Linux звуковая карта правильно определилась и работает, материал этого раздела не имеет особой важности для вас, поскольку все необходимое выполнено автоматически.

Некоторые дистрибутивы Linux предоставляют также утилиту для настройки звука, такую как *sndconfig*, которая пытается обнаружить и настроить звуковую карту, обычно при некотором вмешательстве пользователя. Необходимо свериться с системной документацией и, запустив утилиту конфигурирования звуковых устройств, если таковая есть, проверить, работает ли она.

Если у вас более старая карта ISA или ISA PnP либо ваша карта неправильно определилась, потребуется прибегнуть к ручной процедуре, которую мы здесь описываем. Последующие инструкции исходят из предположения, что вы используете драйверы OSS/Free. При работе с ALSA процедура аналогична, но если вы пользуетесь коммерческими драйверами (OSS/4Front или драйвером, предоставленным изготовителем устройства), следует обратиться к документации, поставляемой с драйвером, поскольку процесс может иметь существенные отличия.

В приведенной здесь информации предполагается также, что Linux установлена на машине с архитектурой x86. Поддержка звука существует и на других архитектурах, но не все драйверы поддерживаются, и могут существовать некоторые отличия в именах устройств и т. д.

Сбор информации об имеющейся аппаратуре

Предполагается, что звуковая карта уже установлена на машине. Если это не так, нужно сначала ее установить. Если вы проверили работу звуковой карты с другой операционной системой, установленной на вашей машине, это будет свидетельствовать о том, что проблемы, если они возникнут в Linux, обусловлены программным обеспечением на том или ином уровне.

Нужно определить тип установленной у вас карты, включая изготовителя и модель. Определите тип шины – ISA, ISA PnP или PCI. Если на карте есть перемыч-

ки, следует записать их положение. Если вам известно, какие ресурсы использует карта в данный момент (IRQ, адрес I/O, канал DMA), зафиксируйте и эту информацию.

Если всю эту информацию получить не удалось, не отчаивайтесь. Можно обойтись и без нее, но впоследствии потребуется провести некоторое расследование. Например, на ноутбуках или системах с интегрированной звуковой аппаратурой вам не удастся взглянуть отдельно на звуковую карту.

Настройка ISA Plug and Play (не обязательно)

Современные звуковые карты с шиной PCI не требуют выполнения дополнительной настройки. Более старые звуковые карты с шиной ISA настраивались с помощью перемычек. Карты ISA PnP настраиваются в Linux с помощью утилит ISA Plug and Play. Если вы не уверены, что ваша звуковая карта относится к типу ISA PnP, попробуйте выполнить команду *pnpdump* и посмотреть, нет ли в ее выводе чего-либо, похожего на звуковую карту. Для типичной звуковой карты в выводе должны быть примерно такие строки:

```
# Card 1: (serial identifier ba 10 03 be 24 25 00 8c 0e)
# Vendor Id CTL0025, Serial Number 379791851, checksum 0xBA.
# Version 1.0, Vendor version 1.0
# ANSI string -->Creative SB16 PnP<--
```

Общая процедура конфигурирования устройств ISA PnP следующая:

1. Сохранить файл */etc/isapnp.conf*, если он существует.
2. Сгенерировать файл настроек с помощью команды *pnpdump >/etc/isapnp.conf*.
3. Отредактировать файл, раскомментировав строки с нужными настройками устройства.
4. Выполнить команду *isapnp* для настройки карт Plug and Play (обычно при запуске системы).

Большинство современных дистрибутивов Linux осуществляют инициализацию карт ISA PnP. У вас уже может быть подходящий файл */etc/isapnp.conf*, либо он потребует частичного редактирования.

Дополнительные подробности о настройке карт ISA PnP можно найти на страницах справочного руководства для *isapnp*, *pnpdump* и *isapnp.conf*, а также в документе «ISA Plug and Play HOWTO» из проекта Linux Documentation.

Настройка ядра (не обязательно)

Чаще всего складывается ситуация, когда вы работаете с ядром, полученным во время установки системы, а все звуковые драйверы включаются в виде загружаемых модулей, и тогда в этом шаге нет необходимости.

Возможно, вам потребуется скомпилировать новое ядро. Если нужные вам модули звукового драйвера ядра отсутствуют в ядре, с которым вы работаете в настоящее время, вам придется выполнить эту операцию. Если вы предпочитаете включать драйверы непосредственно в ядро, а не пользоваться загружаемыми модулями, вам также потребуется новое ядро.

Подробное описание порядка пересборки ядра вы найдете в главе 18.

Настройка модулей ядра

В большинстве случаев звуковые драйверы ядра представляют собой загружаемые модули, которые могут динамически загружаться и выгружаться ядром. Вам необходимо обеспечить загрузку правильных драйверов. Сделать это можно с помощью файла настроек, например */etc/conf.modules*. Типичная запись для звуковой карты может выглядеть в нем так:

```
alias sound sb
alias midi opl3
options opl3 io=0x388
options sb io=0x220 irq=5 dma=1 dma16=5 mpu_io=0x330
```

Необходимо указать звуковой драйвер, который должен использоваться, и правильные значения адреса ввода-вывода, IRQ и каналов DMA, записанные вами ранее. Последние настройки требуются только для карт ISA и ISA PnP, поскольку карты PCI могут определять их автоматически. В предыдущем примере, приведенном для 16-разрядной карты Sound Blaster, нам пришлось указать в первой строке драйвер `sb` и задать параметры драйвера в последней строке.

В некоторых системах настройки содержатся в файле */etc/modules.conf* и (или) нескольких файлах в каталоге */etc/modutils*, поэтому за подробным описанием порядка настройки модулей следует обратиться к документации по дистрибутиву Linux. В дистрибутивах Debian для этой цели можно воспользоваться утилитой *modconf*.

Обычно на практике сложность заключается лишь в том, чтобы определить, какой драйвер использовать. Результаты работы *pnpdump* для карт ISA PnP и *lspci* для карт PCI могут помочь установить, какого типа карта у вас установлена. Затем можно поискать сведения о драйвере в документации, перечень которой вы найдете в «Sound HOWTO», или в исходных текстах ядра, которые на Linux-системах обычно находится в каталоге */usr/src/linux/Documentation/sound*.

Пусть, например, для некоторого ноутбука в результатах работы *lspci* сообщается о следующем звуковом устройстве:

```
00:05.0 Multimedia audio controller: Cirrus Logic CS 4614/22/24 [CrystalClear
          SoundFusion Audio Accelerator] (rev 01)
```

Для этой системы подходящим звуковым драйвером будет `cs46xx`. Возможно, потребуется провести некоторые эксперименты, и для верности можно попробовать загрузить различные модули ядра и посмотреть, не обнаруживают ли они звуковую карту.

Тестирование

Первое, что нужно сделать для проверки установки, – это убедиться, что модуль ядра загружен. Это можно сделать с помощью команды *lsmod*: она должна показать, что соответствующий модуль находится в списке загруженных:

```
$ /sbin/lsmod
Module            Size   Used by
parport_pc       21256    1 (autoclean)
lp                6080    0 (autoclean)
parport          24512    1 (autoclean) [parport_pc lp]
```

```
3c574_cs          8324   1
serial           43520   0 (autoclean)
cs46xx           54472   4
soundcore         3492   3 [cs46xx]
ac97_codec        9568   0 [cs46xx]
rtc               5528   0 (autoclean)
```

Здесь представляют интерес драйверы `cs46xx`, `soundcore` и `ac97_codec`. Если драйвер обнаружил карту, ядро должно также записать в журнал сообщение, которое извлекается с помощью команды *dmesg*. Она может выводить много результатов, поэтому лучше использовать команду постраничного вывода, такую как *less*:

```
PCI: Found IRQ 11 for device 00:05.0
PCI: Sharing IRQ 11 with 00:02.0
PCI: Sharing IRQ 11 with 01:00.0
Crystal 4280/46xx + AC97 Audio, version 1.28.32, 19:55:54 Dec 29 2001
cs46xx: Card found at 0xf4100000 and 0xf4000000, IRQ 11
cs46xx: Thinkpad 600X/A20/T20 (1014:0153) at 0xf4100000/0xf4000000, IRQ 11
ac97_codec: AC97 Audio codec, id: 0x4352:0x5914 (Cirrus Logic CS4297A rev B)
```

Для карт ISA информация находится в файле устройства */dev/sndstat*. Однако для карт PCI это не так. Типичный вывод может выглядеть следующим образом:

```
$ cat /dev/sndstat
OSS/Free:3.8s2++-971130
Load type: Driver loaded as a module
Kernel: Linux curly 2.2.16 #4 Sat Aug 26 19:04:06 PDT 2000 i686
Config options: 0

Installed drivers:

Card config:

Audio devices:
0: Sound Blaster 16 (4.13) (DUPLEX)

Synth devices:
0: Yamaha OPL3

MIDI devices:
0: Sound Blaster 16

Timers:
0: System clock

Mixers:
0: Sound Blaster
```

Если вывод выглядит нормально, можно протестировать звуковую карту. Вначале можно запустить программу-микшер и убедиться в том, что устройство микшера обнаруживается, уровни сигналов можно изменять и при этом не возникают ошибки. Нужно узнать, какие программы микширования есть в системе. Часто встречаются такие программы, как *aumix*, xmix и KMix.

Теперь попробуйте воспроизвести какой-нибудь звуковой файл (например, WAV) с помощью программы-плеера и убедитесь в том, что слышите его. В графической среде, такой как KDE или GNOME, должен быть подходящий плеер, в противном случае поищите утилиту командной строки, например *play*.

Если воспроизведение действует, можно проверить запись. Подключите микрофон ко входу mic звуковой карты и запустите программу записи, такую как *rec* или *vrec*. Проверьте, можно ли записать звук в файл WAV и воспроизвести его. Проконтролируйте в настройках микшера выбор правильного устройства ввода и установите приемлемые уровни сигнала.

Можно также проверить правильность воспроизведения файлов MIDI. Некоторые программы-проигрыватели MIDI требуют наличия звуковой карты с ЧМ-синтезатором, другие – нет. Наибольшее распространение получили такие MIDI-плееры, как Playmidi, KMid и KMidi. Тестирование устройств на шине MIDI выходит за рамки данной книги.

Хорошим информационным ресурсом по MIDI и MIDI-устройствам служит сайт *http://midistudio.com*. Официальные спецификации MIDI можно взять у MIDI Manufacturers Association на веб-сайте *http://www.midi.org*.

Видеодрайверы

При настройке ядра вы можете столкнуться с множеством параметров и драйверов, имеющих отношение к видео. В разделе Multimedia Drivers (Драйверы мультимедиа) можно выполнить настройку драйверов Video For Linux, которые реализуют поддержку устройств видеозахвата и радиотюнеров. В категории Graphics Support (Поддержка графики) можно включить поддержку кадрового буфера, имеющегося в различных видеокартах, чтобы приложения могли получить доступ к видеоаппаратуре через стандартный интерфейс кадрового буфера ядра. Дополнительную информацию о порядке сборки ядра вы найдете в главе 18.

X-сервер также требует наличия поддержки видеоаппаратуры. Программное обеспечение оконной системы X, поставляемое в составе дистрибутива, должно включать в себя все драйверы, распространяемые с открытыми исходными текстами. Но кроме этого в составе дистрибутива могут поставляться и закрытые драйверы, распространяемые изготовителями видеокарт. Если такие драйверы отсутствуют, вам придется самостоятельно получить их и установить. За дополнительной информацией об X Window System обращайтесь к главе 16.

Альтернативные устройства ввода

При настройке ядра в разделе Input Device Support (Поддержка устройств ввода) можно включить поддержку различных специализированных драйверов мыши, джойстиков и активных экранов.

В случае сканеров и цифровых камер со стороны ядра достаточно будет обеспечить поддержку того типа интерфейса, который используется устройством (последовательный порт, SCSI, USB и т. д.). Взаимодействие непосредственно с устройством в этом случае выполняется самими приложениями или с помощью библиотек, таких как SANE или *libgphoto2*.

Встраиваемые и другие устройства мультимедиа

Переносные устройства мультимедиа, предназначенные для проигрывания музыки, приобрели очень высокую популярность. В маленьких устройствах в качестве

носителя используются карты флэш-памяти, в более крупных – жесткие диски, имеющие большую емкость. Как правило, такие устройства могут проигрывать музыку в форматах MP3, WAV или WMA Windows. Имеются также специализированные DVD-проигрыватели, предназначенные для просмотра видеофильмов.

Информация на такие устройства записывается с помощью персонального компьютера. В настоящее время большинство из этих устройств официально не поддерживают Linux. Однако устройства, использующие стандартный протокол обмена с запоминающими устройствами USB большой емкости, должны работать с Linux не хуже, чем с любой другой операционной системой. Многие устройства ведут обмен информацией с помощью закрытых протоколов. Для некоторых из них были созданы утилиты под Linux, иногда в результате обратной разработки. Также возможно использовать программное обеспечение, поставляемое производителем для Windows, запуская его в среде Wine. Остается только надеяться, что в будущем производители аппаратных средств будут официально поддерживать Linux.

Окружения рабочего стола

В данном разделе обсуждается поддержка мультимедийных возможностей, предоставляемая двумя основными окружениями рабочего стола KDE и GNOME, которые обсуждались в главе 3. Обратите внимание: эти две среды не являются взаимоисключающими. Вы можете пользоваться приложениями из среды GNOME в среде KDE и наоборот. Разумеется, существуют другие окружения рабочего стола и оконные менеджеры, которые обладают уникальными характеристиками, но KDE и GNOME – это два самых крупных окружения, которые обычно включаются во все наиболее распространенные дистрибутивы.

KDE

Рабочий стол KDE, или K Desktop Environment, рассматривался нами в главе 3. Что касается мультимедиа, KDE предлагает следующие возможности:

- Микшер звука (KMix).
- Запись звука (Krec).
- Различные медиаплееры, поддерживающие возможность воспроизведения звука и видео (Noatun, Juk, Kaboodle, Kaffeine и прочие).
- Проигрыватель компакт-дисков (KsCD).
- Проигрыватель MIDI (KMid).
- Утилита копирования и кодирования звуковых дорожек с аудиодисков (KAudioCreator).
- Инструмент создания звуковых эффектов (*artsbuilder*).

Поскольку все перечисленные приложения входят в состав одного и того же окружения рабочего стола, они тесно интегрированы между собой. Например, веб-броузер KDE – Konqueror – может проигрывать аудио- и видеофайлы, а приложения KDE могут проигрывать звуки, чтобы извещать пользователя о важных событиях.

Поддержка мультимедиа в KDE основана на aRts – аналоговом синтезаторе реального времени. Одна из частей aRts – это сервер звука *artsd*, который обслу-

живает вывод звука так, чтобы обеспечить возможность одновременного воспроизведения звука несколькими приложениями. Сервер звука взаимодействует непосредственно со звуковыми драйверами – OSS или ALSA.

Кроме того, в KDE существует масса мультимедийных приложений, которые официально не являются частью KDE из-за того, что они пока не обладают достаточно высокими качественными характеристиками либо существуют как самостоятельные проекты. Первые можно найти в разделе *kdenonbeta* проекта KDE. Ссылки на последние обычно можно найти с помощью индексных сайтов, таких как *http://freshmeat.net* или *http://www.kde-apps.org*.

GNOME

GNOME – это еще один свободно распространяемый проект рабочего стола, описанный в главе 3. Как и KDE, GNOME может предложить пользователям микшер звука, возможность записи звука, проигрыватель компакт-дисков и различные медиапроигрыватели. Поддержка мультимедиа встроена в Nautilus, файловый менеджер среды GNOME. В качестве сервера звука в GNOME используется *esd*, который предоставляет возможность совместного использования звуковых устройств несколькими приложениями одновременно.

При работе в смешанной среде, когда одновременно могут работать приложения KDE и GNOME, может возникать проблема, которая состоит в том, что серверы звука конфликтуют между собой при использовании звуковых ресурсов. К моменту написания этих строк разработчики обоих проектов выражали неудовлетворенность реализациями своих серверов звука и обсуждали возможность совместной разработки замены, которая могла бы совместно использоваться приложениями рабочих столов KDE и GNOME. Это наконец позволило бы одновременно запускать приложения KDE и GNOME без возникновения конфликтов.

Совместимость с Windows

Проект Wine – это технология, позволяющая запускать приложения Windows под управлением Linux. Более подробно о Wine будет рассказываться в главе 28. Некоторые коммерческие мультимедийные приложения могут работать в среде Wine.

Коммерческая версия Wine от CodeWeavers под названием CrossOver поддерживает множество мультимедийных приложений, включая Adobe Photoshop, Apple iTunes, Windows Media Player и модули расширения к веб-броузерам для проигрывания форматов QuickTime, Flash и ShockWave.

Компания TransGaming Technology предлагает продукт под названием Cedega, который оптимизирован для работы игровых программ Windows, требующих поддержки DirectX. Он основан на альтернативной версии Wine, известной как ReWind, которая имеет менее ограничительные условия лицензионного соглашения по сравнению с Wine.

Некоторые мультимедийные приложения, такие как MPlayer, в довесок к Wine способны напрямую использовать некоторые библиотеки Windows (*.dll*), обеспечивающие поддержку патентованных кодеков.

Мультимедийные приложения

После настройки аппаратного обеспечения в Linux можно будет воспользоваться некоторыми мультимедийными приложениями, которых так много, что просто невозможно было бы перечислить здесь все, поэтому мы опишем лишь общие категории программ и перечислим некоторые наиболее популярные приложения. Отыскать все эти приложения вы сможете с помощью ссылок, которые приводятся в конце главы. Там же вы найдете подробные описания некоторых популярных или особенно практичных приложений.

Далее будут рассматриваться следующие категории мультимедийных приложений:

- Программы микширования звука, позволяющие настраивать уровни записи и воспроизведения.
- Медиапроигрыватели аудио- и видеофайлов и дисков.
- Инструменты записи на CD и DVD, позволяющие создавать аудио- и видеодиски.
- Синтезаторы речи, обладающие возможностью распознавания и синтеза речи.
- Инструменты редактирования графических изображений, звука и видео, позволяющие создавать и обрабатывать файлы мультимедиа.
- Программы записи звука, позволяющие создавать звуковые файлы.
- Инструменты сочинения музыкальных произведений, позволяющие создавать файлы форматов MIDI и MP3.
- Интернет-телефония и средства организации конференц-связи по компьютерным сетям.
- Модули расширения к броузерам, позволяющие отображать мультимедийные данные внутри веб-броузера.

Микшеры звука

Микшеры звука позволяют изменять уровни выходных и входных сигналов на звуковой карте. По своей сути микшеры мало чем отличаются друг от друга. Вообще, при работе в KDE или GNOME наилучшим выбором будет использование микшера, распространяемого в составе окружения рабочего стола, который обычно проявляет себя в виде изображения динамика на панели задач. Микшеры звука, запускаемые из командной строки, такие как *aumix*, обычно бывает удобно использовать из сценариев или стартовых файлов для установки желаемых уровней в процессе загрузки системы или при работе вне графической среды, например при входе в удаленную систему.

На рис. 9.1 изображено окно программы KMix – микшера, распространяемого в составе KDE.

Медиаплееры

Медиаплееры – это наиболее обширная категория приложений с самым широким диапазоном возможностей и пользовательских интерфейсов. Ни одно из приложений не в состоянии удовлетворить все потребности. Некоторые из при-

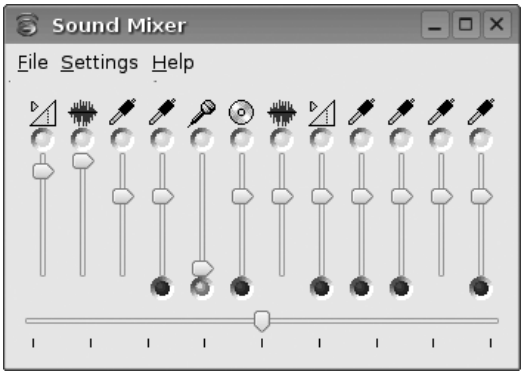


Рис. 9.1. *KMix*

ложений разрабатывались как легковесные и быстрые программы, тогда как другие преследовали цель предложить как можно больше возможностей. Например, даже в пределах окружения рабочего стола KDE существует с полдюжины различных плееров.

Если вы работаете в среде KDE или GNOME, то наверняка у вас уже имеется по крайней мере один медиаплеер. Если это действительно так, рекомендуем попользоваться этим плеером хотя бы первое время, так как он, скорее всего, без проблем будет взаимодействовать с сервером звука и обеспечит наилучшую интеграцию с окружением рабочего стола.

При выборе медиаплеера обращайте внимание на следующие его особенности:

- Поддержка различных драйверов (например, OSS или ALSA) или серверов звука (KDE aRts и GNOME *esd*).
- Привлекательный внешний вид. Многие плееры предоставляют возможность изменения внешнего вида с помощью оболочек, благодаря чему можно загрузить и установить альтернативные варианты оформления пользовательского интерфейса.
- Поддержка списков музыкальных произведений, что позволяет создавать и сохранять списки с любимыми произведениями.
- Различные аудиоэффекты, такие как графический эквалайзер, расширение стереоэффекта, реверберация, удаление голоса, а также визуальные эффекты, сопровождающие воспроизведение музыки.
- Поддержка файлов разных форматов, таких как CD, WAV и видеоформатов.

Далее следует список из некоторых наиболее популярных плееров:

Xmms

Один из популярных медиапроигрывателей с интерфейсом пользователя по умолчанию, аналогичным Winamp. Если он отсутствует в вашем дистрибутиве, его можно загрузить с сайта *http://www.xmms.org*. Внешний вид проигрывателя приводится на рис. 9.2.

Рис. 9.2. *Xmms*

Рис. 9.3. *Xine*

Xine

Xine – это полнофункциональный аудио- и видеоплеер, поддерживающий возможность воспроизведения файлов самых разных форматов и протоколов потокового медиа. Сайт проекта: *http://xine.sourceforge.net*. Внешний вид проигрывателя приводится на рис. 9.3.

MPlayer

MPlayer – это еще один популярный видеоплеер, поддерживающий широкий диапазон форматов файлов и позволяющий загружать кодеки из библиотек Windows. Поддерживает возможность вывода на различные устройства как с помощью X11, так и прямо на видеокарту. Сайт проекта: *http://www.mplayer.hu*.

Из-за проблем с лицензированием MPlayer обычно не включается в состав дистрибутивов Linux и потому должен загружаться отдельно.

Инструменты записи CD и DVD

Если вы пользуетесь KDE или GNOME, базовые возможности записи данных на компакт-диски вы найдете в файловых менеджерах. Если вам необходимо нечто большее или требуется помощь, чтобы пройти весь процесс записи, тогда можно воспользоваться специализированными приложениями.

Обратите внимание: многие приложения записи компакт-дисков, обладающие графическим интерфейсом, используют утилиты командной строки, такие как *cdrecord* и *cdrdao*, с помощью которых выполняется извлечение аудиотреков, создание образов ISO и запись на компакт-диск. Для достижения максимальной гибкости некоторые опытные пользователи предпочитают работать непосредственно с этими инструментальными средствами.

X-CD-Roast

Одним из первых приложений с графическим интерфейсом, предназначенных для записи компакт-дисков, была программа X-CD-Roast. Она до сих пор остается достаточно функциональной и надежной, хотя появились и другие

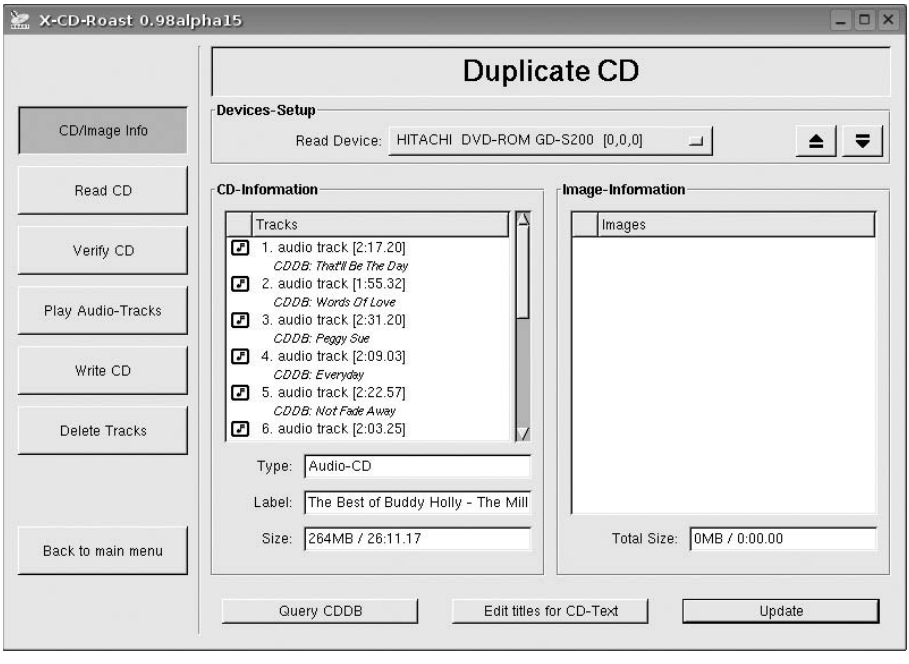


Рис. 9.4. *X-CD-Roast*

приложения, способные предложить более интуитивный интерфейс на основе мастеров. Внешний вид программы приводится на рис. 9.4.

K3b

K3b – это популярный инструмент записи компакт-дисков для рабочего стола KDE. Его интерфейс файлового менеджера напоминает программы записи компакт-дисков для Windows, как, например, Easy CD Creator. Внешний вид программы приводится на рис. 9.5. Введение в работу с этой программой вы найдете в разделе «Запись компакт-дисков с помощью K3b» главы 3.

Gcombust

Gcombust – это программа записи компакт-дисков с графическим интерфейсом, построенным на базе библиотек Gtk. Домашняя страница проекта: *http://www.abo.fi/~jmunsin/gcombust*. Внешний вид программы приводится на рис. 9.6.

Распознавание и синтез речи

Синтез и распознавание речи необходимы приложениям, предназначенным для людей с ограниченными возможностями, а также для специализированных применений, например телефонии, где возможен только аудиоконтакт.

Устройства синтеза речи делятся на два основных типа. Специализированные аппаратные синтезаторы выпускаются в виде отдельных периферийных устройств и выполняют функцию преобразования текста в речь. Главное их преимущество – они разгружают компьютер, принимая на себя все обязанности по

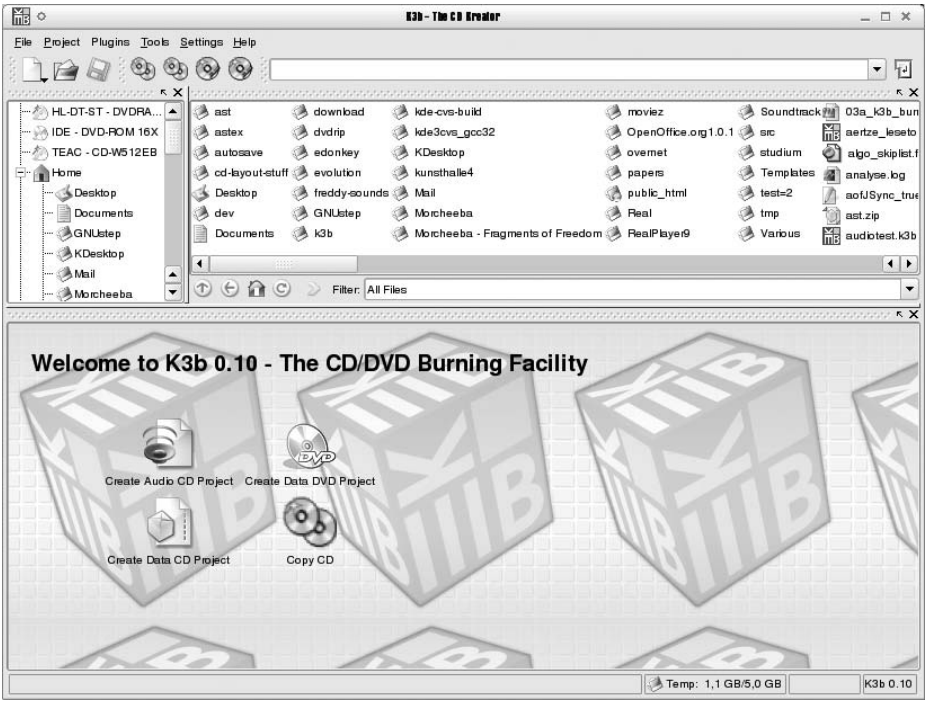


Рис. 9.5. *K3B*

выполнению речевого преобразования, и, кроме того, способны выполнять свои функции более качественно. Программные синтезаторы работают непосредственно на персональном компьютере. Как правило, они имеют более низкую стоимость, нежели аппаратные решения, но увеличивают нагрузку на центральный процессор и иногда, если используется бесплатное программное обеспечение, дают слишком низкое качество речи.

Rsynth

Пакет Rsynth предоставляет простую утилиту командной строки с именем say, которая выполняет преобразование текста в речь. Этот пакет входит в состав большинства дистрибутивов Linux.

Emacspeak

Emacspeak – это окружение речевого рабочего стола, предназначенного для пользователей со слабым зрением. Он предоставляет возможность считывания текстовой информации с экрана и передачи ее программному или аппаратному синтезатору речи. Подробную информацию о проекте можно найти на сайте *http://www.cs.cornell.edu/home/raman/emacspeak*.

Festival

Festival – это программная платформа, предназначенная для строительства систем синтеза речи. Она поддерживает возможность произнесения слов на различных языках и может встраиваться в системы, разработка которых ве-

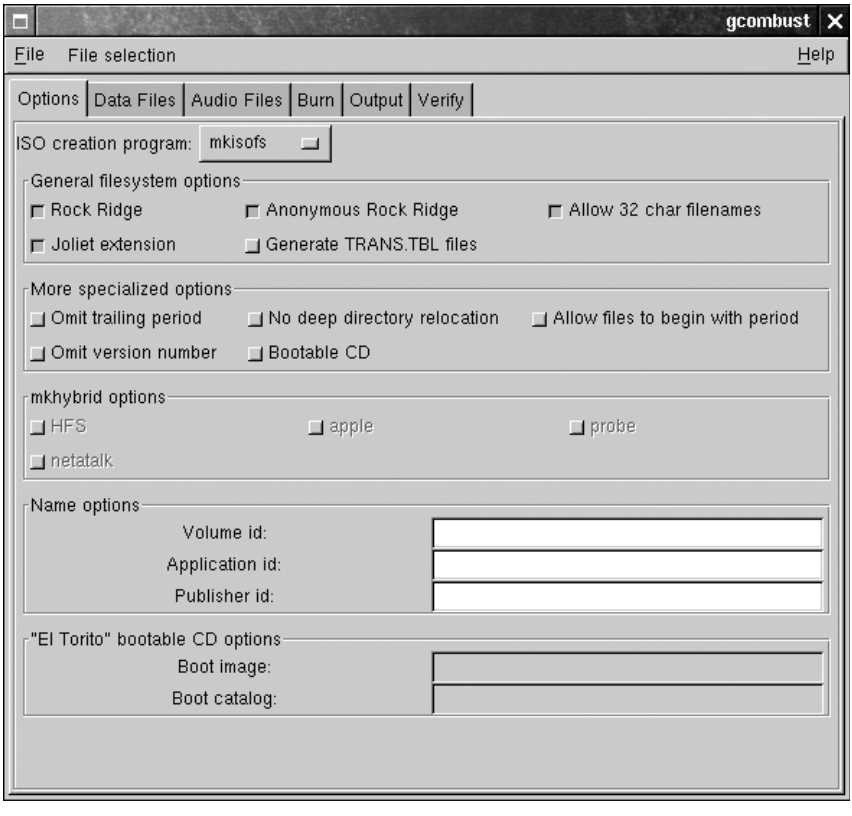


Рис. 9.6. *Gcombust*

дется на языках программирования C++, Java, Scheme и даже на языке командной оболочки. Домашняя страница проекта находится по адресу *http://www.cstr.ed.ac.uk/projects/festival*.

IBM ViaVoice

Компания IBM предлагает версию ViaVoice для Linux – комплект инструментальных средств для разработки речевых систем, который предоставляет возможность как синтеза, так и распознавания речи. Это коммерческий (то есть не бесплатный) продукт.

Инструменты редактирования и управления изображениями, звуком и видео

В этом разделе описываются популярные средства редактирования файлов с изображениями, видео и звуком, а также организации коллекций изображений.

The GIMP

Название «The GIMP» происходит от GNU Image Manipulation Program (программа GNU манипулирования изображениями). Эта программа предназначена для решения таких задач, как ретуширование фотографий, создание новых

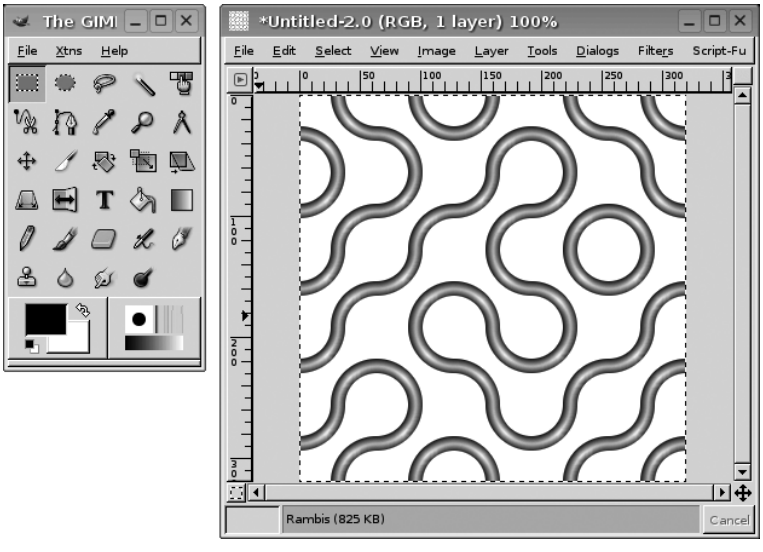


Рис. 9.7. *GIMP*

изображений и изменение существующих. Проект активно развивается на протяжении последних нескольких лет, в результате программа получилась очень стабильной, с богатейшим набором возможностей. На рис. 9.7 приводится внешний вид программы. Официальный веб-сайт GIMP: *http://www.gimp.org*.

CinePaint

Программа CinePaint, ранее известная под названием Film Gimp, является графическим редактором, предназначенным для ретуширования изображений с высокой разрешающей способностью. Она широко используется в киноиндустрии для создания фоновых изображений и покадрового ретуширования фильмов. Программа CinePaint основана на GIMP и обладает рядом дополнительных возможностей редактирования видеоизображений, такими как увеличенная до 128 бит глубина цвета, упрощенный способ навигации между кадрами, и поддерживает форматы видеофайлов Kodak Cineon, ILM OpenEXR, Maya IFF и 32-разрядный TIFF. Внешний вид программы приводится на рис. 9.8. Веб-сайт CinePaint: *http://www.cinepaint.org*.

Gphoto2

Gphoto2 – набор приложений для работы с цифровыми камерами в Linux и других UNIX-подобных системах. Включает библиотеку *libgphoto2*, которая поддерживает почти 400 моделей цифровых камер. Среди других основных компонентов: *gphoto2*, программа командной строки для доступа к цифровым камерам, и графическое приложение Gtkam. Домашняя страница проекта: *http://www.gphoto.org*. Внешний вид программы Gtkam приводится на рис. 9.9.

Digikam

Digikam – это приложение, предназначенное для работы с цифровыми камерами в KDE. Для взаимодействиями с камерами использует библиотеку *libgphoto2*. Внешний вид программы приводится на рис. 9.10.

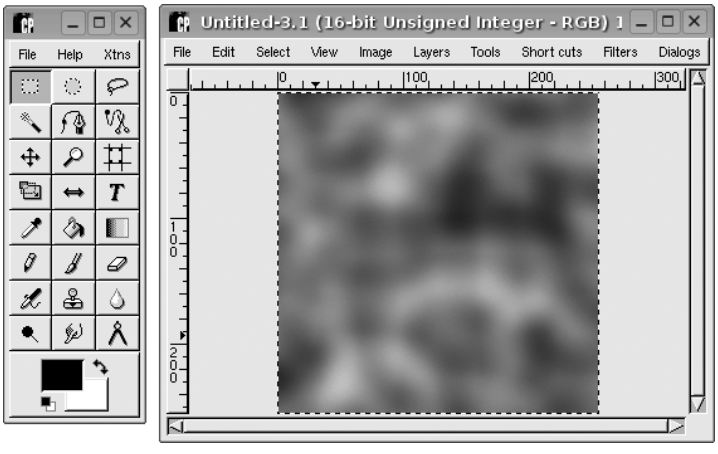


Рис. 9.8. *CinePaint*

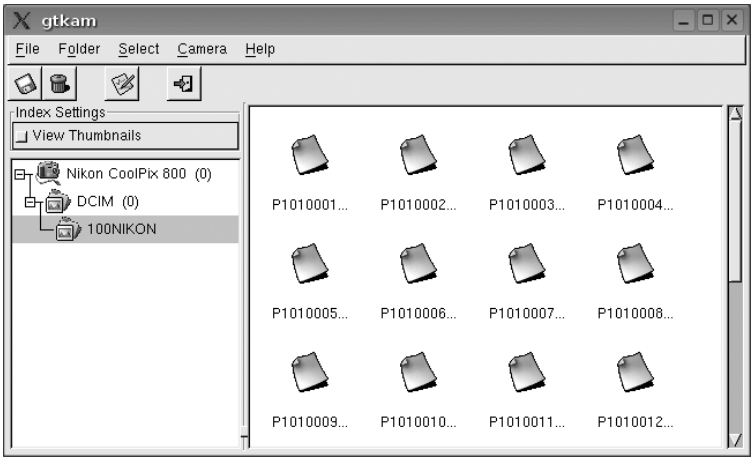


Рис. 9.9. *Gtkam*

Kooka

Kooka – это программа сканирования изображений для KDE. Поддерживает работу со сканерами через библиотеку SANE. Помимо простого сканирования изображений программа обладает возможностью распознавания текстовой информации с помощью разнообразных модулей OCR (Optical Character Recognition – оптическое распознавание символов). Внешний вид программы приводится на рис. 9.11.

Инструменты для работы с изображениями

Для просмотра, редактирования изображений и организации их в виде коллекций существует большое разнообразие программ. В этой главе мы рассмотрим некоторые из них.

Рис. 9.10. *Digikam*

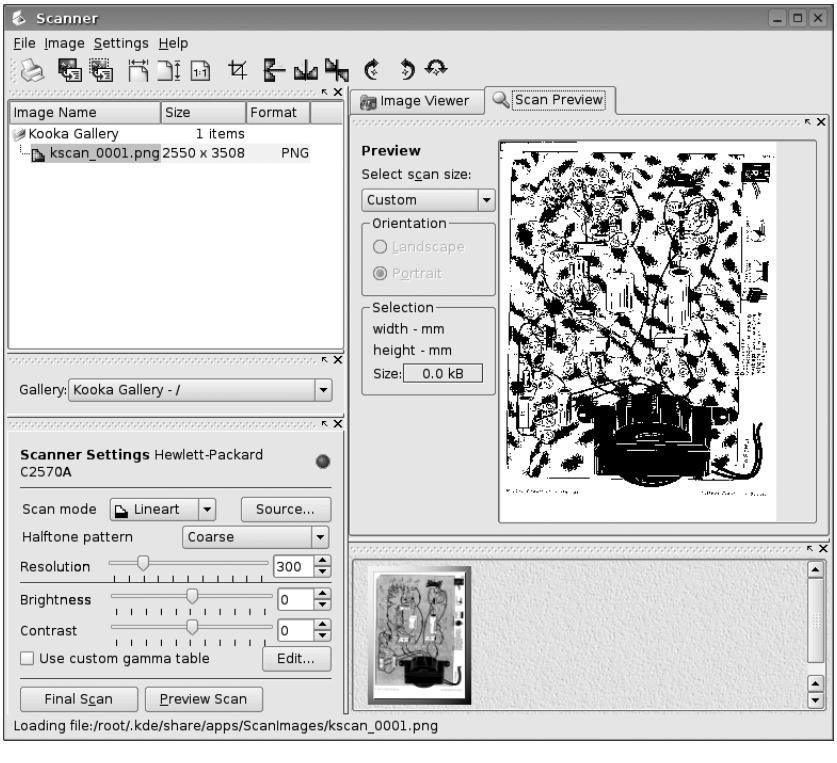


Рис. 9.11. *Kooka*

Организация коллекций изображений с помощью KimDaBa

Существует огромное количество приложений, предназначенных для просмотра изображений, и, исходя из нашего опыта, все они могут быть сгруппированы в две основных категории: те, что способны создавать страницы HTML из наборов изображений, и те, что позволяют просматривать изображения в режиме поочередной демонстрации. Количество приложений, входящих в состав каждой категории, исчисляется сотнями, если не тысячами, главным образом эти приложения отличаются лишь характеристиками, которые можно отнести на счет *личных предпочтений* или даже *религии*. Выбрать понравившееся вам приложение вы сможете на одном из сайтов, распространяющих программное обеспечение для Linux. В этом разделе мы остановимся на приложениях, предназначенных для различных целей.

Приложение KimDaBa (KDE Image DataBase – база данных изображений для KDE) прекрасно характеризуется следующей цитатой, взятой с сайта проекта:

> Если по складу своего характера вы походите на меня, наверняка у вас имеются сотни или даже тысячи изображений, накопившиеся со времен приобретения вашей первой фотокамеры. Некоторые из этих изображений были получены обычной камерой, другие – цифровой. В течение долгих лет вы наивно полагали, что сможете всю жизнь помнить историю появления каждой своей фотографии, всех людей, что были запечатлены на них, и, безусловно, сможете вспомнить дату создания каждого снимка.
>
> Лично я понял, что это совершенно невозможно, и потому мне потребовался инструмент, с помощью которого я смог бы организовать хранение не только цифровых, но и обычных фотографий, который помог бы мне описать мои снимки и отыскивать нужные среди других фотографий. Именно для этих целей и предусматривалась программа KimDaBa.

Основная идея, заложенная в KimDaBa, – отнести каждое изображение к определенной категории, где его потом проще будет отыскать, и присвоить ему ключевое слово (которое позднее могло бы быть использовано для поиска). Эти категории могут использоваться для выбора и просмотра изображений. На рис. 9.12 приводится внешний вид броузера KimDaBa.[1]

Поиск изображений ведется следующим образом: в списке наверху (см. рис. 9.12) имеется несколько категорий, такие как Keywords (Ключевые слова), Locations (Расположения), Persons (Персоны) и т. д. Чтобы отыскать нужную фотографию, например, где изображен ваш знакомый, вы выбираете щелчком мыши категорию Persons (Персоны) и в появившемся списке выбираете имя своего знакомого. После этого вы вернетесь в первый экран, со списком категорий. Однако теперь программа сузит круг доступных изображений, ограничив его заданным именем, то есть KimDaBa будет отображать информацию только о тех фотографиях, в описании которых встречается заданное вами имя. Если число отобранных изображений оказалось не слишком большим, то можно просто выбрать категорию View Images (Просмотр изображений) и отыскать нужную фотографию визуально. В противном случае можно повторить процесс поиска. Например, если вы хотите

[1] Существует возможность добавлять свои собственные категории, если имеющиеся в KimDaBa не устраивают вас по каким-либо причинам.

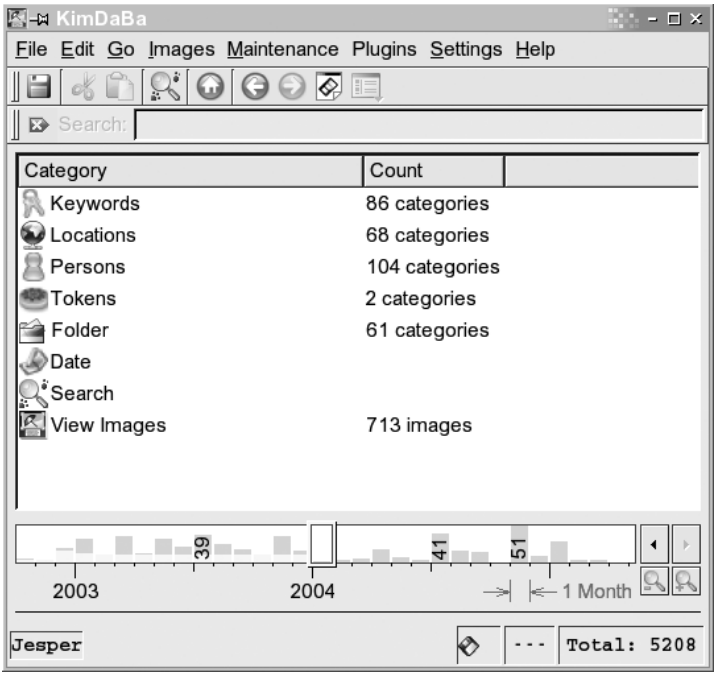


Рис. 9.12. *Поиск изображений в KimDaBa*

отыскать фотографии, где изображен ваш друг с супругой, то просто выберите категорию Persons (Персоны) снова, но на этот раз выберите из списка имя его жены. Если вам нужно отыскать фотографии друга в некотором городе, выберите категорию Locations (Расположения) и укажите название местоположения.

Бесплатный сыр, как известно, бывает только в мышеловке. В случае с KimDaBa это означает, что вам придется упорядочить все ваши изображения и отнести их к той или иной категории, а это может оказаться достаточно трудоемким делом, если у вас имеются тысячи фотографий. Однако именно для этих целей и создавалась программа KimDaBa, один из ее основных критериев – возможность хранения информации о десятках и даже сотнях тысяч фотографий.

Существует два способа категоризовать изображения в KimDaBa, выбор которых зависит от того, на чем вы сосредоточили свое внимание, но сначала заметим, что при наличии времени задача категоризации может быть решена шаг за шагом.

Первый способ заключается в том, чтобы выбрать одно или более изображений в окне предварительного просмотра (которое открывается выбором пункта View Images (Просмотр изображений)) и затем вызвать контекстное меню щелчком правой кнопки мыши. В контекстном меню следует выбрать пункт Configure Images One at a Time (Настроить изображения по одному) (комбинация клавиш Ctrl+1) или Configure All Images Simultaneously (Настроить все изображения сразу) (комбинация клавиш Ctrl+2).

Функция Configure All Images Simultaneously (Настроить все изображения сразу) позволяет указать местоположение всех изображений, скажем Лас Вегас, всего не-

сколькими щелчками мыши, тогда как функция Configure Images One at a Time (Настроить изображения по одному) даст возможность обойти все изображения последовательно и указать, например, кто на них изображен.

На рис. 9.13 приводится диалог, в котором указываются свойства изображений. В этом диалоге можно выбрать элементы из списков или начать вводить название в поле ввода, в последнем случае по мере ввода KimDaBa будет предлагать возможные варианты. (На рис. 9.13 я ввел лишь символ J, а программа KimDaBa отыскала первое совпадение в списке.)

Как вариант, определять свойства изображений можно в процессе просмотра (например, в режиме демонстрации серии снимков). В этом режиме изображение просто связывается с некоторым символом нажатием соответствующей клавиши. Данная возможность предназначена для исправления описаний, например, когда при просмотре фотографий вы вдруг вспомнили, что забыли указать имя своего друга, присутствующего на фотографии. Позднее эти символы можно будет использовать для поиска и просмотра фотографий, так же как используются имена людей, местоположения и ключевые слова. В этом случае можно просто отобрать фотографии с установленной меткой-символом и затем с помощью первого метода, описанного выше, дополнить описание фотографии.

Как только все фотографии будут описаны, вы сможете отбирать их для просмотра самыми разными способами. Не такой уж и редкий пример использования KimDaBa: вы сидите со своей подругой на диване в гостиной, вспоминаете, как весело провели каникулы на Мальорке в 2000 году, и решаете еще раз просмот-

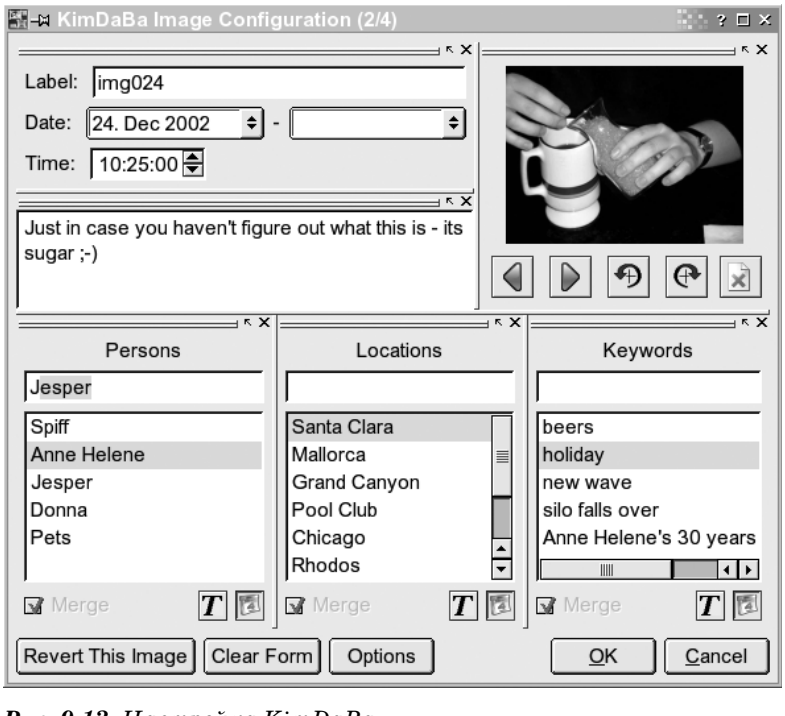


Рис. 9.13. *Настройка KimDaBa*

реть фотографии на ноутбуке. Вы вводите ключевую фразу «Каникулы Мальорка 2000» и начинаете просматривать фотографии в режиме демонстрации.

В процессе просмотра вдруг встречается фотография, сделанная по прибытии домой, где запечатлен ваш старый друг, с которым вы не встречались долгое время. Не выходя из режима просмотра, вы щелкаете на ссылке с его именем (вся информация, которую вы вводили, будет выводиться в поле информации во время просмотра), и на экране появляется окно броузера KimDaBa, круг доступных изображений в котором будет ограничен заданным именем. Теперь с помощью поля ввода даты можно ограничить перечень доступных изображений только теми, которые были сделаны в период с 1990 по 2000 годы. В этот список могут попасть фотографии, сделанные на вечеринке много лет тому назад, и снова щелчком на ссылке можно перейти к просмотру фотографий, сделанных на этой вечеринке. Такие просмотры нередко заканчиваются далеко за полночь.

Использование программы GIMP для работы с изображениями

Введение. GIMP (GNU Image Manipulation Program – программа GNU манипулирования изображениями) предназначена для решения таких задач, как ретуширование фотографий, создание новых изображений и изменение существующих. Проект активно развивается на протяжении последних нескольких лет, в результате программа получилась очень стабильной, с богатейшим набором возможностей.

Официальная домашняя страница GIMP: *http://www.gimp.org*. Справочное руководство в электронном виде доступно по адресу *http://docs.gimp.org*, а дополнительные модули, расширяющие возможности программы, можно найти на сайте *http://registry.gimp.org*.

Программу GIMP можно использовать как простой графический редактор растровых изображений, но главная ее сила – именно манипулирование изображениями. В этой книге мы расскажем о некоторых инструментах и методиках работы с изображениями. Чтобы полностью описать все возможности, которыми обладает GIMP, потребовалось бы выпустить целую книгу, поэтому рассматривайте данное описание как толчок к более глубокому знакомству с GIMP.

К моменту написания этих строк текущей была версия GIMP 2.2. В последующих версиях наверняка появятся некоторые незначительные отличия в функциональных возможностях и в пользовательском интерфейсе, но основная идея приложения останется неизменной.

Инструменты выделения. После запуска программы на экране появляется *окно с набором инструментов*, как показано на рис. 9.14. В верхней половине окна находятся кнопки, каждая из которых представляет определенный инструмент. В этом же окне имеется полоса меню, содержащая пункты для создания новых изображений, загрузки, сохранения, изменения параметров настройки и тому подобного. Ниже области с кнопками располагается раздел, где отображается текущий цвет фона и переднего плана, тип выбранной кисти и тому подобное. В самом нижнем разделе окна отображаются параметры текущего инструмента.

Чтобы создать новое изображение, необходимо выбрать пункт меню File (Файл) →New (Создать). В результате будет открыто новое окно с пустым изображением, где можно будет поэкспериментировать с разными инструментами.

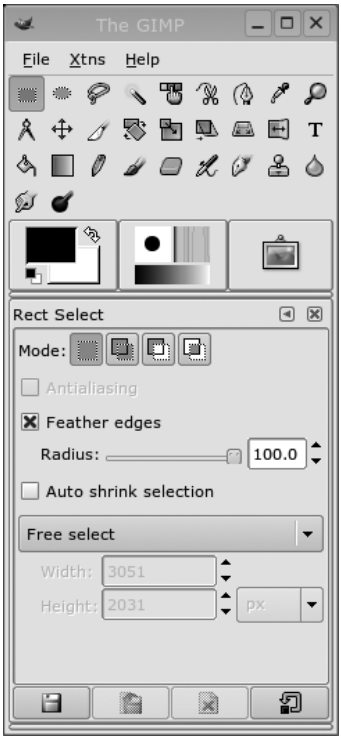


Рис. 9.14. *Палитра инструментов GIMP*

Первые пять инструментов – это инструменты выделения: прямоугольной области, эллиптической области, произвольной области, «волшебная палочка», по цвету и форме. *Выделение* – это область изображения, с которой работают практически все инструменты и фильтры GIMP, таким образом, область выделения – это очень важное понятие. Текущая область выделения обозначается границами в виде бегущей пунктирной линии, которую можно скрыть или показать с помощью комбинации клавиш Ctrl+Z.

Первые три инструмента выбора, за исключением выделения области произвольной формы, очень похожи друг на друга. В процессе выделения прямоугольной или эллиптической области есть возможность сохранить неизменным отношение сторон прямоугольника или осей эллипса, удерживая клавишу Shift. В окне параметров инструмента выделения можно определить режим выделения: добавить к текущему выделению, вычесть из текущего выделения, заменить текущее выделение и создать выделение из пересечения с текущим.

Все инструменты выделения характеризуются величиной растушевки краев – этот параметр определяет, насколько плавным будет переход на границах выделенной области. Пример использования растушевки приводится на рис. 9.15.

Инструмент «Волшебная палочка» позволяет выбрать некоторую точку на изображении и выделить непрерывную область вокруг этой точки того же цвета.

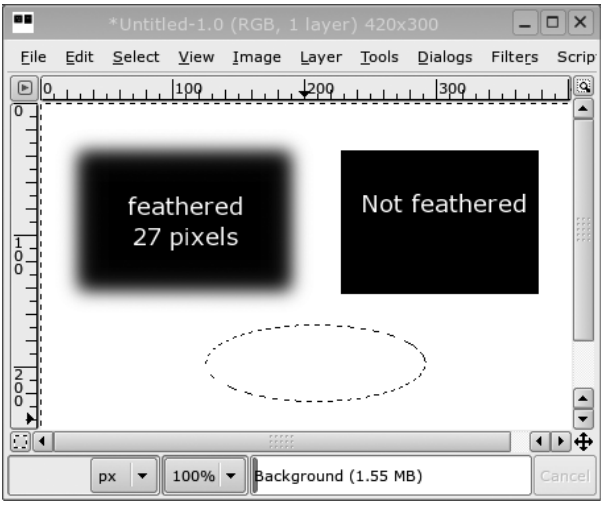


***Рис. 9.15**. Выделения в GIMP*

Степень подобия цвета можно изменять с помощью движка. Выделение по цвету в чем-то напоминает инструмент «Волшебная палочка» с той лишь разницей, что инструмент выделения по цвету производит выделение *всех* пикселов с подобным цветом независимо от того, образуется ли при этом непрерывная область выделения. Наконец, инструмент выбора форм позволяет разместить на изображении множество точек и попробовать соединить их кривыми, которые образуют края области выделения. Когда в изображении будет установлено достаточно много точек, чтобы выделить контуром требуемую область, просто щелкните мышью в середине этой области, и полученная ранее кривая превратится в выделение.

Инструменты рисования и стирания. Рисовать изображения в GIMP можно с помощью таких инструментов, как Карандаш (Pencil), Кисть (Paintbrush), Аэрограф (Airbrush) и Перо (Ink). Они отличаются друг от друга формой рисования: Карандаш дает линии с четкими границами, при рисовании Кистью границы получаются размытыми, Аэрограф дает полупрозрачные линии, Перо при медленном перемещении дает толстые линии, а при быстром – тонкие.

Чтобы заполнить выделенную область некоторым цветом, используются инструменты Заливка (Paintbucket) и Заливка градиентом (Gradient fill). Выбор стиля заливки, цвета и (или) вида градиента производится щелчком мыши на элементах управления в средней части окна с набором инструментов.

Некоторые пользователи испытывают сложности с рисованием прямых линий в GIMP, но, так как в ваших руках сейчас находится эта книга, вы узнаете, как это делается: выберите один из инструментов рисования, поместите указатель мыши в точку, где должно находиться начало линии, и при нажатой клавише Shift переместите указатель мыши в точку, где линия должна оканчиваться, после чего щелкните один раз левой кнопкой мыши. Теперь можно продолжить рисование другого отрезка из этой точки или отпустить клавишу Shift и полюбоваться нарисованной вами прямой линией.

Если в процессе рисования будет допущена ошибка, воспользуйтесь быстрой комбинацией Ctrl+Z, которая откатывает сделанные изменения. Откат может производиться не только для последнего изменения, откатить можно несколько изменений. Помимо инструментов рисования в GIMP имеются инструменты стирания, позволяющие стирать выделенные пикселы.

Все манипуляции с инструментами рисования ограничиваются областью выделения, если таковая имеется.

Инструменты ретуширования фотографий. Инструменты, описываемые в этом разделе, главным образом используются для внесения незначительных (а порой и существенных) изменений в цифровые фотографии. Инструмент Штамп (Clone) очень удобно использовать для удаления пятен с фотографий. Работает он следующим образом: щелчок левой кнопкой мыши с удерживаемой клавишей Ctrl определяет начальную точку исходной области, а затем рисование где-нибудь в другом месте изображения будет выполняться «копиями» исходной области. На рис. 9.16 показана фотография, где в правом верхнем углу видна часть крыши, случайно попавшая в кадр. Слева изображена исходная фотография, а справа – фотография после того, как из нее был удален нежелательный фрагмент с помощью инструмента Штамп.

Последний инструмент в палитре – Осветление/Затемнение (Burn/Dodge). Он используется для осветления или затемнения участков изображения, что позволяет корректировать тени и блики на фотографиях.

Выравнивание цветов. На заключительных стадиях изменения фотографий иногда бывает необходимо скорректировать яркость, контрастность или баланс цветов изображения. Для этих целей в GIMP существуют несколько инструментов. Доступ к ним осуществляется через пункт контекстного меню Colors (Цвет).

Один из наиболее замечательных инструментов – инструмент коррекции цветовых уровней. Он позволяет скорректировать уровень контрастности элементов изображения. Фотография, которая приводится на рис. 9.17, была сделана при

Рис. 9.16. *Инструмент Штамп в GIMP*

существенном недостатке освещения. В результате она получилась малоконтрастной и размытой.

Попробуем исправить недостаток с помощью диалога Levels (Уровни)! Откройте диалог Levels (Уровни), выбрав пункт контекстного меню Colors (Цвет)→Levels (Уровни). В данном случае диалог имеет вид, показанный на рис. 9.18.

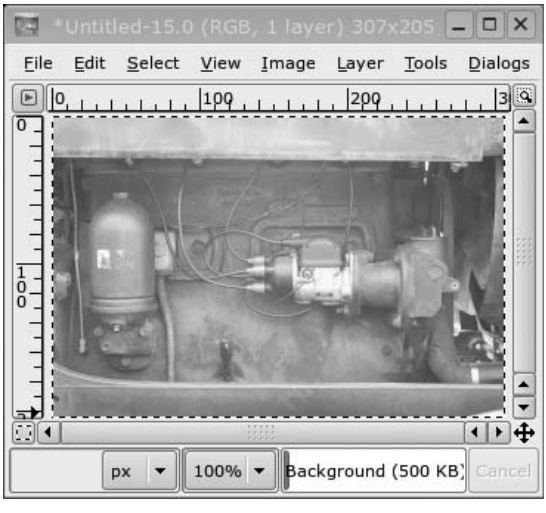


Рис. 9.17. *Исходная фотография*

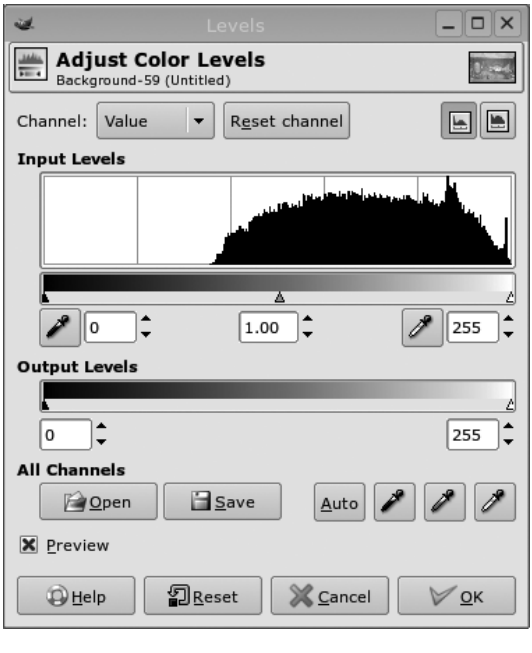


Рис. 9.18. *Диалог Levels (Уровни)*

Диаграмма, расположенная под надписью Input Levels (Уровни на входе), является гистограммой значений яркости изображения. Левый край гистограммы отвечает за черный цвет, а правый – за белый. Здесь видно, что первые 40% гистограммы не заполнены – это означает, что мы впустую тратим полезный динамический диапазон. Ниже гистограммы находится шкала с тремя треугольными движками. Черный и белый движки отвечают за установку самого черного и самого белого элементов в изображении, а серый предназначен для регулирования распределения значений в этих пределах. Чтобы устранить размытость изображения, черный движок можно передвинуть, как показано на рис. 9.19. Результат приводится на рис. 9.20.

Изменить контраст можно с помощью инструментов Brightness-Contrast (Яркость-Контраст) и Curves (Кривые). Первый из них очень прост и содержит всего два движка, с помощью которых регулируются яркость и контраст. Второй инструмент дает намного больше возможностей. На рис. 9.21 показана оригинальная фотография и две измененные версии с различными кривыми. Для среднего изображения была применена кривая увеличения контраста (рис. 9.22), а к правому – кривая уменьшения контраста (рис. 9.23). Кривые описывают отображение значений пиксела на самого себя. Прямая линия, расположенная под углом 45 градусов, описывает идентичное отображение; любое изменение положения линии будет отражаться на изображении. Наилучшие результаты могут быть получены при незначительных отклонениях от первоначальной линии.

Баланс цвета можно настроить с помощью нескольких инструментов, таких как Color Balance (Цветовой баланс) и Hue-Saturation (Тон-Насыщенность). Инструменты Levels (Уровни) и Curves (Кривые) также могут использоваться для коррекции отдельных составляющих цвета с целью достижения различных эффектов. Но су-

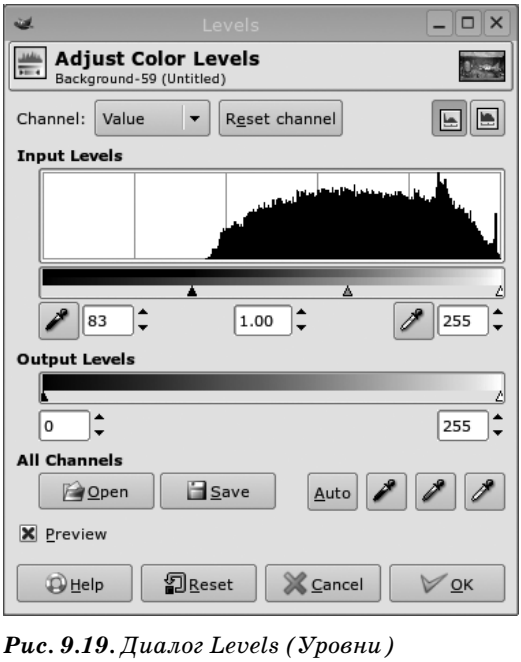


Рис. 9.19. *Диалог Levels (Уровни)*

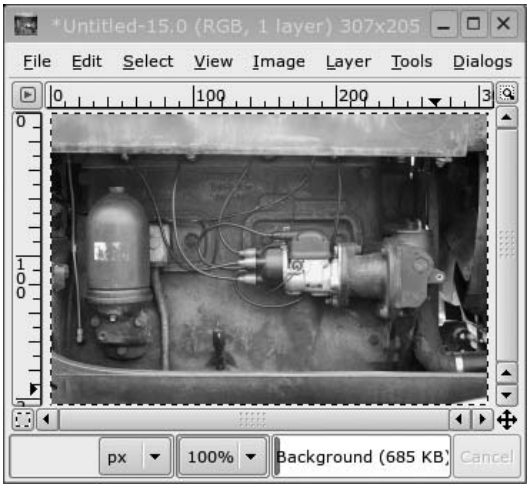


Рис. 9.20. *Фотография после выравнивания уровней*

Рис. 9.21. *Коррекция цветовыми кривыми*

ществует еще один инструмент – Channel Mixer (Микшер каналов). В отличие от других, данный инструмент находится в разделе контекстного меню Filter (Фильтры)→Colors (Цвета), пункт Channel Mixer (Микшер каналов). Микшер каналов может использоваться для взвешенного объединения каналов цвета (красный, зеленый и синий) в каждом из выходных каналов. Иногда эта процедура бывает необходима для преобразования цветных изображений в одноцветные, поскольку порой она дает лучшие результаты по сравнению с уменьшением насыщенности цветов. На рис. 9.24 показан внешний вид диалога Channel Mixer (Микшер каналов), а на рис. 9.25 – две одноцветные версии одного и того же цветного изображения. Верхнее изображение было получено в результате обычного умень-

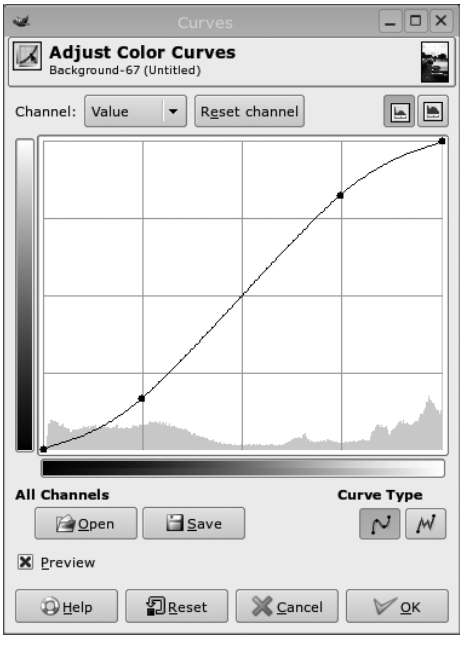


Рис. 9.22. *Кривая увеличения контраста*

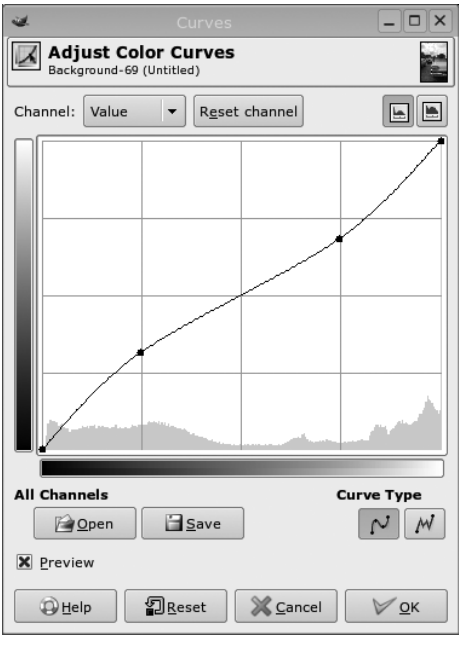


Рис. 9.23. *Кривая уменьшения контраста*

шения насыщенности, а нижнее основано на канале синего цвета, и похоже, что выделяется птица, а не фон. При преобразовании цветного изображения в одноцветное желательно проверить каждую составляющую цвета. Подробнее об этом будет рассказываться в следующем разделе.

Слои и каналы. Проще всего получить доступ к слоям и каналам через комбинированное окно, объединяющее слои, каналы, контуры и историю отмены действий. Для этого нужно щелкнуть правой кнопкой мыши на изображении и выбрать пункт контекстного меню Dialogs (Диалоги)→Create New Dock (Создать новую панель)→Layers, Channels & Paths (Слои, каналы и контуры). Слои и каналы дают возможность управлять различными характеристиками изображения простым и понятным способом.

Каналы

Изображения состоят из одного или более каналов. Полноцветные изображения имеют три цветовых канала: один для красного, один для зеленого и один для синего цвета. Индексированные изображения и изображения, построенные в градациях серого цвета, состоят из единственного канала. Изображения всех типов могут иметь дополнительный альфа-канал, который описывает степень прозрачности изображения (белый цвет – непрозрачный, черный – абсолютно прозрачный). Щелкая на кнопках с изображением глаза, можно включать и выключать отдельные каналы, выстраивая изображение на основе подмножества каналов. Манипуляции с изображениями производятся только на выбранном подмножестве каналов. В обычном режиме выбираются

Рис. 9.24. *Микшер каналов*

все каналы, но если необходимо внести изменения, например, только в канал красной составляющей, следует снять выделение со всех остальных каналов. В этом случае все операции рисования будут воздействовать только на красный канал. Такой прием может использоваться для устранения эффекта «красных глаз». Существует возможность добавлять к изображению дополнительные каналы с помощью кнопок внизу диалога. Кроме того, имеется возможность сохранять области выделения в виде отдельных каналов, а также преобразовывать каналы в области выделения. Такой прием позволяет «запомнить» множество областей выделения для последующего использования и расширяет возможности выделения. На рис. 9.26 показана вкладка Channels (Каналы) в комбинированном окне Layers, Channels & Paths (Слои, каналы и контуры). Зеленый и синий каналы – видимы, из них только зеленый канал выбран для редактирования.

Слои

Слои – это чрезвычайно мощная абстракция, имеющаяся в GIMP. Представить слои можно как стопку из нескольких изображений, расположенных одно над другим. Если некоторый слой имеет альфа-канал, то уровни ниже его будут проглядывать сквозь прозрачные области вышележащего слоя. Управлять уровнем прозрачности слоя можно с помощью движка Transparent (Прозрачность) (рис. 9.27). С помощью кнопок в нижней части диалога можно создавать, копировать, удалять и перемещать слои вверх или вниз. Слоям

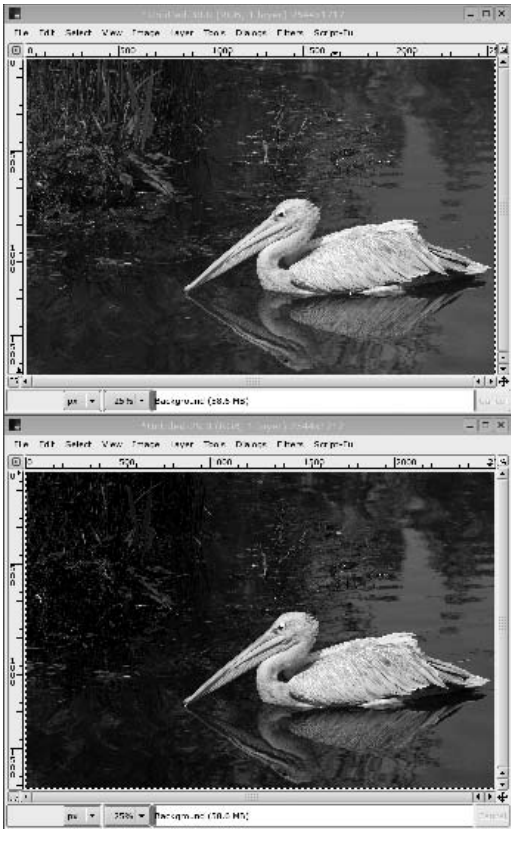


Рис. 9.25. *Пример микширования каналов*

можно назначать имена, для этого нужно щелкнуть правой кнопкой мыши на слое и выбрать пункт контекстного меню Edit Layer Attributes (Правка атрибутов слоя). На рис. 9.27 показано содержимое вкладки Layers (Слои) в окне Layers, Channels & Paths (Слои, каналы и контуры) для изображения, состоящего из трех слоев, два из которых являются копиями первоначального фона.

Вернемся к изображению с автомобилем, которое приводилось в примере с кривыми. Трава, деревья и сам автомобиль наиболее эффектно смотрятся с более высоким контрастом, а вот небо при увеличении контраста теряет детали и лучше выглядит при пониженном контрасте, потому что в этом случае детали на ярком небе прорисовываются гораздо четче. Давайте попробуем совместить, казалось бы, несовместимое. Оставим самый нижний слой в покое – он будет служить в качестве образца. Средний слой назовем Ground (Земля), а самый верхний – Sky (Небо). Теперь оставим видимым только слой Ground (Земля), для чего выделим его, и с помощью инструмента Curves (Кривые) увеличим контраст изображения. Затем выберем слой Sky (Небо) и, наоборот, уменьшим контраст. Теперь нам нужно объединить два слоя. Для этого добавим *маску слоя* к самому верхнему слою, но прежде нам нужно выделить об-

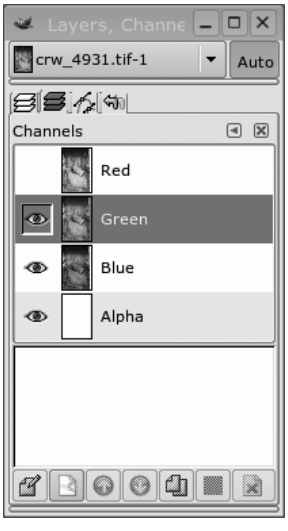


Рис. 9.26. *Диалог Channels (Каналы)*

Рис. 9.27. *Диалог Layers (Слои)*

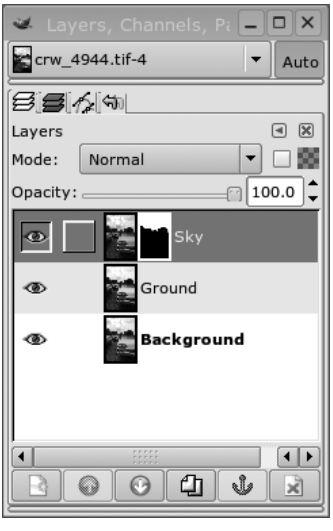


Рис. 9.28. *Слои и маска*

ласть будущей маски. С помощью инструмента «Волшебная палочка» постарайтесь выделить как можно большую часть неба так, чтобы в область выделения не попали ни земля, ни деревья, ни автомобиль. Напомним, что щелчок мышью с удерживаемой клавишей Shift добавляет новую область к существующему выделению. Когда большая часть неба окажется выделенной, щелкните правой кнопкой мыши на самом верхнем слое в диалоге и выберите пункт контекстного меню Add Layer Mask (Добавить маску слоя), а затем в появившемся диалоге установите флажок Selection (Выделение). Не волнуйтесь, если у вас не получилось выделить область с точностью до пиксела, – мы можем исправить этот недостаток позднее. Нажмите комбинацию клавиш Ctrl+Shift+A, чтобы снять текущее выделение – нам оно больше не нужно. Теперь окно Layers, Channels & Paths (Слои, каналы и контуры) должно напоминать рис. 9.28.

Щелкая на том или ином значке в слое Sky (Небо), можно выбирать сам слой или его маску для редактирования. Выберите маску уровня и увеличьте масштаб изображения на границе между деревьями и небом. Теперь маску можно откорректировать, просто рисуя пером черного или белого цвета. Белый цвет будет проявлять небо, а черный – деревья. Чтобы вместо изображения увидеть одну только маску, щелкните правой кнопкой мыши на маске в диалоге Layers, Channels & Paths (Слои, каналы и контуры) и выберите пункт контекстного меню Show Mask (Показать маску слоя). Конечный результат должен выглядеть примерно так, как показано на рис. 9.29.

До сих пор мы использовали слои только в обычном режиме, но существуют и другие режимы. Дополнительные режимы оказывают влияние на взаимодействие с нижележащими слоями, причем весьма интересными способами. Например, можно заставить пикселы слоя затемняться, осветляться, умножаться и т. д. со значениями пикселов нижележащего слоя. В умелых руках это может дать впечатляющие эффекты. На рис. 9.30 демонстрируется изображение из предыду-

Рис. 9.29. *Два слоя*

щего примера, к которому сверху был добавлен новый прозрачный слой. Этот новый слой содержит результат выделения прямоугольной области, немного меньшей, чем само изображение, с большим радиусом растушевки по краям (10% высоты изображения). Область выделения была инвертирована и залита черной краской. В качестве режима слоя был выбран режим Overlay (Затемнитель), что вызвало эффект затемнения нижележащих слоев вдоль границ. Это создает эффект фотографии, сделанной с помощью старой или дешевой камеры, и придает сцене особое настроение. Если бы вместо затемнения мы использовали обычный режим, эффект был бы слишком сильным и выглядел бы неестественно. Пробуйте самостоятельно экспериментировать с различными режимами!

Фильтры. Последняя основная функциональная возможность GIMP, которую мы опишем, – это фильтры. Фильтры – это эффекты, которые могут накладываться как на все изображение, так и на область выделения. В состав GIMP входит большое количество разнообразных фильтров, а кроме того, существует возможность дополнять этот набор фильтрами, расширяющими возможности GIMP. Фильтры находятся в разделе контекстного меню Filters (Фильтры). Мик-

Рис. 9.30. *Два слоя*

шер каналов – это одна из разновидностей фильтров. Мы рассмотрим здесь два фильтра: Gaussian Blur (Гауссово размывание) и Unsharp Mask (Нерезкая маска), которые наложим на изображение из предыдущего примера.

Гауссово размывание

Этот фильтр дает хороший гладкий эффект размывания. Попробуйте наложить этот фильтр с различными значениями радиуса пятна. В большинстве случаев тип размывания IIR, на наш взгляд, выглядит немного лучше, чем тип размывания RLE.

В нашем примере мы будем размывать само изображение. Мы лишь попробуем разгладить переход между слоями с высоким и низким контрастом. Для этого выберем маску слоя в слое Sky (Небо) и наложим фильтр Gaussian Blur (Гауссово размывание). Радиуса пятна в 8 пикселов будет вполне достаточно. Увеличьте масштаб изображения на границе неба и деревьев и не бойтесь экспериментировать – у вас всегда есть возможность нажать комбинацию клавиш Ctrl+Z, чтобы откатить изменения и повторить попытку. На рис. 9.31 крупным планом показан фрагмент изображения до и после размывания мас-

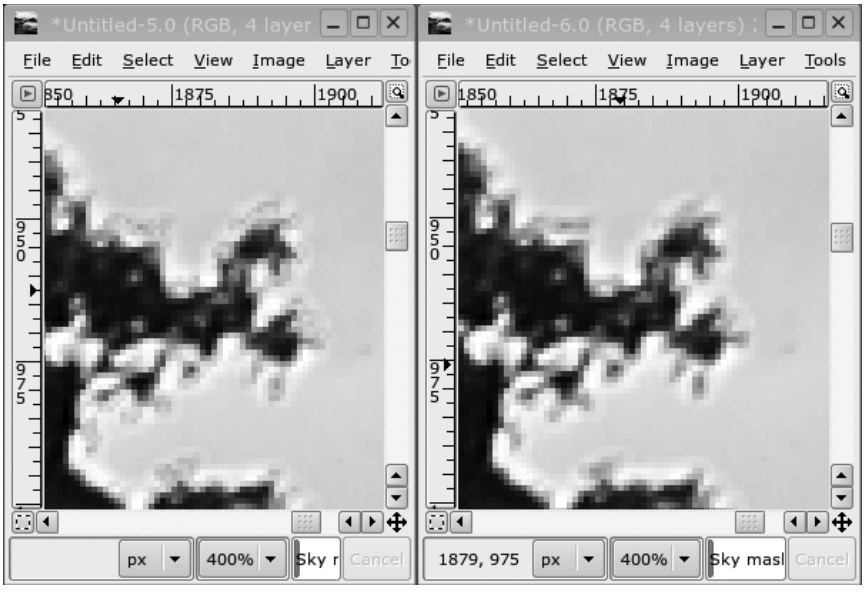


Рис. 9.31. *Нерезкая маска – до и после*

ки. Эффект получается малозаметным, но он очень важен при необходимости сгладить резкие переходы между слоями.

Нерезкая маска

Несмотря на свое название, фильтр Unsharp Mask (Нерезкая маска) предназначен как раз для увеличения ощущения резкости изображения. Он предоставляет большую широту выбора и зачастую обеспечивает лучший результат, нежели более простой фильтр Sharpen Filter (Повышение резкости).

Фильтр Unsharp Mask (Нерезкая маска) работает следующим образом: сначала создается внутренняя копия вашего изображения, на которую накладывается фильтр Gaussian Blur (Гауссово размывание). Затем вычисляются различия между оригиналом и переработанной копией изображения для каждого пиксела, разница умножается на некоторый множитель и добавляется к оригинальному изображению. Идея состоит в том, чтобы размыть резкие переходы чуть больше, чем остальные поверхности, и проявить различия тем сильнее, чем ближе они расположены к границам цветовых переходов в изображении. Добавление различий лишь подчеркивает границы переходов. Параметр Radius (Радиус) нерезкой маски – это радиус пятна гауссова размывания. Параметр Amount (Величина) – это множитель, а параметр Threshold (Порог) определяет величину различий, которые будут игнорироваться фильтром. Выбор более высокого значения порога поможет ослабить шум на цифровых фотографиях.

Взгляните на наш пример с небом и автомобилем еще раз. Обратите внимание: высококонтрастная часть снимка потеряла некоторые детали в тенях после увеличения контраста. Этот недостаток легко можно исправить с помощью фильтра Unsharp Mask (Нерезкая маска). Для этого мы наложим фильтр Unsharp Mask (Нерезкая маска) с большим радиусом пятна и с небольшим зна-

Рис. 9.32. *Два прохода фильтра Unsharp Mask (Нерезкая маска)*

чением параметра Amount (Величина). Эта методика называется локальным увеличением контраста. Сначала создайте копию изображения комбинацией клавиш Ctrl+D и в копии объедините слои, для чего нужно выбрать в контекстном меню пункт Image (Изображение)→Flatten Image (Свести изображение). Затем изображение желательно отмасштабировать по размеру экрана. Для этого откройте диалог масштабирования, выбрав пункт контекстного меню Image (Изображение)→Scale Image (Масштабировать), установите оптимальные размеры и выберите алгоритм интерполяции Bicubic (best) (Кубическое (лучшее)). Теперь можно приступать к наложению фильтра Unsharp Mask (Нерезкая маска) для локального увеличения контраста. На наш взгляд, наилучшие результаты получаются с параметрами Radius (Радиус) – 25, Amount (Величина) – 0,15 и Threshold (Порог) – 0.

В заключение можно немного повысить контраст переходов. Для этого нужно наложить фильтр Unsharp Mask (Нерезкая маска) с маленьким радиусом (0,5), с более высокой величиной (0,5) и с порогом 6. На рис. 9.32 приводится результат наложения нерезкой маски: слева – исходное изображение, в центре – после локального увеличения контраста, и справа – после последнего прохода увеличения резкости.

Инструменты записи

Если у вас появится желание создавать собственные MP3-файлы, вам обязательно понадобится программа-кодировщик. Кроме того, существуют программы, которые помогут вам извлекать дорожки с музыкальных компакт-дисков.

Чисто технически существует возможность выполнять кодирование в формате MP3 с помощью свободно распространяемых инструментальных средств, но требования некоторых патентов придают такой деятельности весьма сомнительный характер. В качестве альтернативы можно рассматривать формат Ogg Vorbis, который определенно свободен от проблем, связанных с патентами. Для работы

с этим форматом необходимо, чтобы его поддерживала программа-проигрыватель, потому что он не совместим с форматом MP3. Однако большинство MP3-плееров, таких как Xmms, уже поддерживают Ogg Vorbis. В отдельных случаях существуют прямые эквиваленты (например, `ogg123` для `mpg123`). Для записи видео в формате Ogg был разработан кодек Ogg Theoris, который является свободно распространяемым и не обременен никакими патентами.

В этом разделе будут перечислены некоторые популярные инструментальные средства с графическим интерфейсом, способные выполнять запись и редактировать файлы мультимедиа.

Krec

В качестве стандартного приложения записи звука в состав KDE входит программа Krec. Она позволяет записывать звук из любого входа микшера, например с микрофона или с компакт-диска, и сохранять запись в виде звукового файла. Программа дает возможность накладывать некоторые звуковые эффекты, но основная ее цель – простая запись звука. Внешний вид программы приводится на рис. 9.33.

Audacity

Audacity – это аудиоредактор, который может выполнять запись, воспроизводить звуковые файлы, читать и записывать звук в наиболее распространенных форматах. Программа дает возможность редактировать аудиофайлы с использованием операций копирования, вырезания и вставки. Позволяет микшировать дорожки и накладывать различные эффекты. Может в графическом виде отображать содержимое аудиофайлов разных форматов.

Внешний вид программы приводится на рис. 9.34. Домашняя страница проекта: *http://audacity.sourceforge.net*.

Ardour

Ardour – это полноценная студия цифровой звукозаписи, предназначенная для замены аналоговых или цифровых ленточных магнитофонов. Программа предоставляет возможность создания многоканальных и многодорожечных записей, микширования, редактирования и наложения эффектов. Внешний вид программы приводится на рис. 9.35. Домашняя страница проекта: *http://ardour.org*.

Рис. 9.33. *Krec*

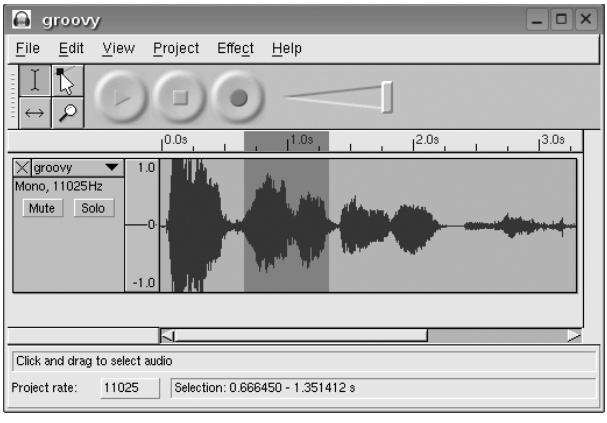


Рис. 9.34. *Audacity*

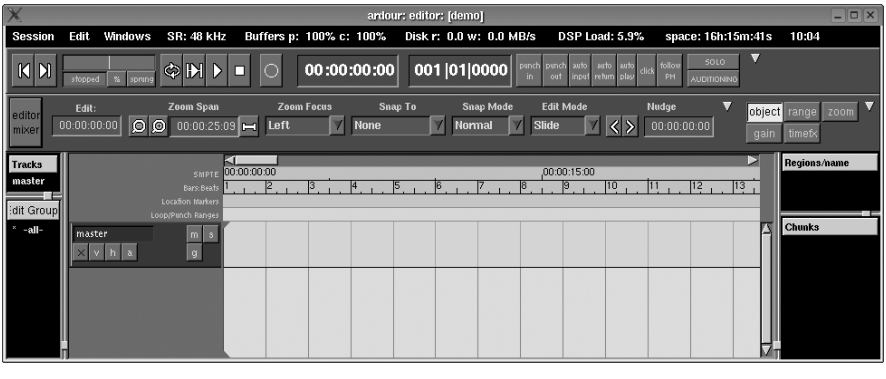


Рис. 9.35. *Ardour*

Freevo

Freevo – это свободно распространяемая платформа для развертывания домашнего кинотеатра, основанная на Linux и открытых аудио- и видеоинструментальных средствах. Программа может проигрывать аудио- и видеофайлы самых популярных форматов. Может использоваться в качестве видеомагнитофона, управляемого с помощью телевизора и дистанционного пульта, или обычного рабочего стола, управляемого монитором и клавиатурой. Внешний вид программы приводится на рис. 9.36. Домашняя страница проекта: *http://freevo.sourceforge.net*.

MythTV

MythTV – это приложение персонального видеомагнитофона, который обладает следующими функциональными возможностями:

- Просмотр и запись телепередач.
- Просмотр изображений.
- Просмотр и захват видеоизображения с дисков DVD.

Рис. 9.36. *Freevo*

- Проигрывание музыкальных файлов.
- Отображение прогноза погоды, новостей и просмотр веб-страниц в Интернете.
- Интернет-телефония и видеоконференции.

Внешний вид программы приводится на рис. 9.37. Домашняя страница проекта: *http://mythtv.org*.

Инструменты сочинения музыки

В помощь композиторам было разработано множество приложений.

MIDI-синтезаторы позволяют композитору редактировать и проигрывать музыку в формате MIDI. А поскольку формат MIDI основан на таких понятиях, как ноты, дорожки и музыкальные инструменты, этот формат часто рассматривается как наиболее естественный способ сочинения музыкальных произведений, нежели непосредственное редактирование цифровых аудиофайлов.

Программы озвучивания нот позволяют композиторам работать с традиционным способом записи музыки и воспроизводить набранные ноты. Иногда программы предоставляют поддержку дополнительных способов записи музыки – в виде табулатур для гитары и других инструментов.

Некоторые программы сочетают в себе возможности MIDI-синтезатора и программы проигрывания нот или могут работать с различными стандартизированными форматами файлов для записи музыкальных произведений.

Brahms

Brahms – это приложение, основанное на KDE, с функциями MIDI-синтезатора, позволяющее композитору редактировать дорожки и воспроизводить их.

Рис. 9.37. *MythTV*

Работает как в терминах формата MIDI, так и в системе традиционной записи нот, для чего используются различные окна редактора. Внешний вид программы приводится на рис. 9.38. Домашняя страница проекта: *http://brahms.sourceforge.net*.

Rosegarden

Rosegarden – это аудио- и MIDI-синтезатор, нотный редактор и многоцелевая среда сочинения и редактирования музыкальных произведений. Позволяет работать как с форматом MIDI, так и с обычной записью нот. Может интегрироваться с другими приложениями рабочего стола KDE. Интерфейс программы переведен примерно на 10 языков.

Внешний вид программы приводится на рис. 9.39. Домашняя страница проекта: *http://www.rosegardenmusic.com*.

LilyPond

LilyPond – это программа для сочинения небольших музыкальных произведений посредством записи нот. В качестве описания использует файлы на языке высокого уровня. Поддерживает множество способов записи музыки, включая аккорды, ударные инструменты, басы, высокие ноты, гитарные табулатуры, современные способы записи (кистевая нотация и ритмическая группировка), тремоло, комбинации (вложенные) в произвольных пропорциях и многое другое.

Входной язык основан на текстовом описании музыки и может быть интегрирован в форматы LaTeX, HTML и Texinfo, что дает возможность создавать документы, содержащие ноты и обычный текст в виде единственного файла. Из этих документов можно получать файлы форматов PostScript и PDF (через TeX), а также файлы формата MIDI.

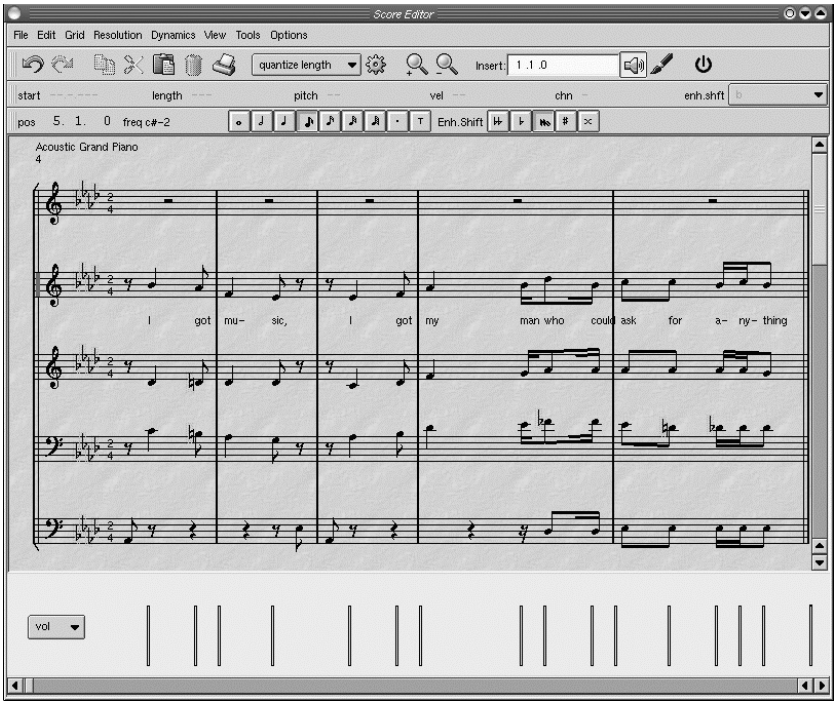


Рис. 9.38. *Brahms*

Домашняя страница проекта: *http://lilypond.org*. Для программы LilyPond имеется оболочка с графическим интерфейсом, которая называется Denemo.

Интернет-телефония и видеоконференции

Телефония через Интернет совсем недавно получила широкое распространение. Передача аудиоинформации производится по локальной сети или через Интернет с помощью технологии VOIP (Voice Over IP – передача голоса по протоколу IP). Установление сеанса передачи мультимедиа (не только аудио) производится в соответствии со стандартом SIP (Session Initiation Protocol – протокол инициирования сеанса). Для обмена аудиоинформацией может использоваться звуковая карта с микрофоном или специализированное устройство, напоминающее традиционный телефон. Интернет-телефония имеет массу преимуществ, но главное из них – это цена; на сегодняшний день многие пользователи имеют постоянное высокоскоростное подключение к Интернету, которое может использоваться для соединения с любым другим пользователем, находящимся в любой точке земного шара и обладающим соответствующим программным обеспечением. При наличии подходящего шлюза можно даже делать телефонные звонки с телефона VOIP на обычный телефон.

Для Linux существует множество приложений VOIP. Одно из самых популярных – KPhone. Помимо передачи аудиоинформации оно позволяет обмениваться

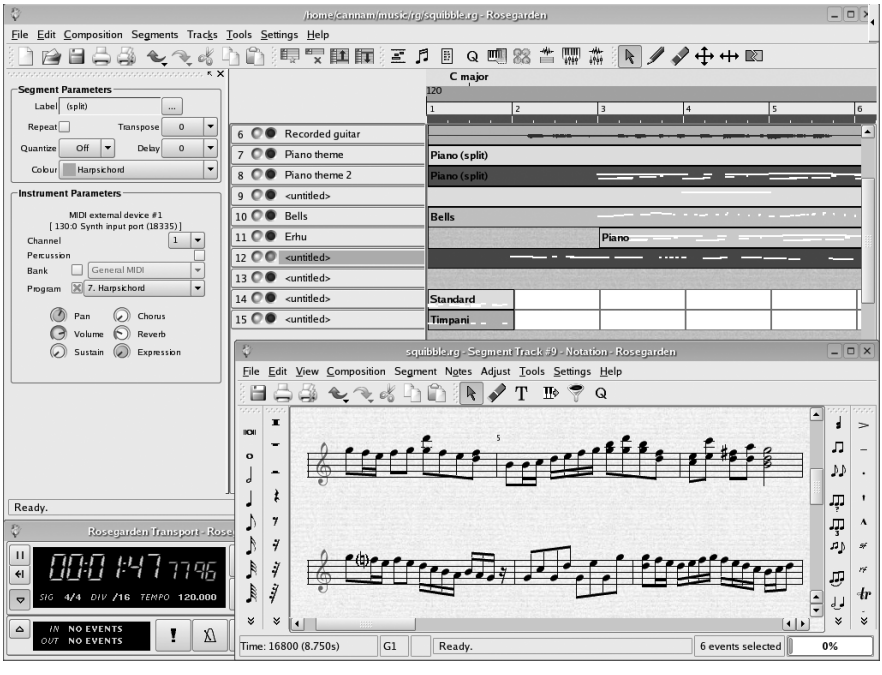


Рис. 9.39. *Rosegarden*

мгновенными сообщениями и имеет поддержку возможности передачи видеоинформации. Домашняя страница проекта: *http://www.wirlab.net/kphone*.

Кроме того, существуют и коммерческие приложения, которые используют закрытые протоколы или их закрытые расширения. Например, проект Skype, в рамках которого бесплатно предоставляется клиентское программное обеспечение, но для выполнения звонков на обычные телефоны необходимо оформлять платную подписку на услуги шлюза. Найти Skype можно на сайте *http://www.skype.com*.

Стандартом проведения видеоконференций в локальных сетях считается протокол H.323. Этот протокол поддерживается приложением Microsoft NetMeeting, который включается в состав операционной системы Microsoft Windows. Для Linux также существуют приложения, совместимые с протоколом H.323, самое известное из них – это GnomeMeeting. Домашняя страница проекта: *http://www.gnomemeeting.org*.

Модули расширения для броузеров

Модули расширения броузеров реализуют возможности отображения веб-броузерами типов данных, отличных от HTML. Некоторые из них квалифицируются как мультимедиа. Они могут быть поделены на три категории:

- Модули расширения, поставляемые вместе с броузером или распространяемые самими производителями броузеров (например, Mozilla или Firefox).

- Модули расширения от сторонних производителей, например Adobe Acrobat, которые могут распространяться бесплатно, но их исходные тексты могут быть закрыты.
- Модули расширения для операционной системы Windows, которые можно запускать в Linux под управлением CrossOver компании CodeWeaver (технология Wine). В данную категорию попадают такие расширения, как Apple QuickTime, Windows Media Player и Adobe Shockwave. Многие из подобных расширений не имеют реализаций специально для Linux.

Формат расширений Netscape поддерживается броузерами компании Netscape, Mozilla и некоторыми другими броузерами, основанными на исходных текстах Mozilla, такими как FireFox. Данный формат расширений также поддерживается броузером Konqueror проекта KDE.

Подведение итогов

В этой главе было рассказано о множестве мультимедийных приложений. Несмотря на то, что каждое отдельно взятое приложение обладает узконаправленной специализацией, при умелом их сочетании можно решать гораздо более сложные задачи. Рассмотрим пример из реальной жизни.

Мне нравится коллекционировать и реставрировать старые ламповые радиоприемники производства 1930–1950 годов. После того как я приводил очередной радиоприемник в рабочее состояние, мне всегда хотелось послушать его. Но когда я включал его и слышал из динамика репортаж с местных спортивных состязаний, мне это казалось неуместным. Разве не было бы здорово услышать какую-нибудь старую радиопостановку, записанную в ту эпоху, когда появился радиоприемник?

Из Интернета можно бесплатно загрузить множество записей старых радиопередач. Многие из них я загрузил на свой компьютер. Кроме того, некоторые записи старых радиопередач были приобретены мною на компакт-дисках. У меня есть даже старые виниловые пластинки и магнитофонные ленты с записями радиопостановок. С помощью Audacity, проигрывателя пластинок и магнитофона, подключенных к входу моей звуковой карты, я преобразовал эти записи в звуковые файлы. Затем, после простейшей обработки с целью устранения посторонних шумов, эти файлы были преобразованы в формат MP3.

Когда у меня возникает желание прослушать их на своем компьютере, я запускаю программу Juk, с помощью которой навожу порядок в своей коллекции, классифицируя записи по типам и степени предпочтения, и часами наслаждаюсь непрерывной музыкой или радиопостановками. Я могу переписывать эти файлы в MP3-проигрыватель, чтобы иметь возможность прослушивать их, когда компьютер мне недоступен, или записать файлы на компакт-диск и воспроизводить их с помощью карманного проигрывателя компакт-дисков.

Используя AM-передатчик малой мощности, я вполне легально могу транслировать радиопередачи в пределах своего дома. Недорогой AM-передатчик можно свободно приобрести в любом месте, подключить его к выходу звуковой карты или проигрывателя компакт-дисков и транслировать у себя дома старинные радиопередачи в диапазоне длинных волн. Теперь, когда я включаю старый радиоприемник, я могу слушать Берна и Вуди Аллена, *The Shadow* или свинг 1940-х

годов. Я могу даже убедить легковерных посетителей в том, что старые радиоприемники способны принимать старые радиопередачи.

Инструменты и средства разработки мультимедийных приложений

KDE и GNOME уже обсуждались выше. Они обеспечивают поддержку графики и звука, которая может использоваться не слишком требовательными мультимедийными приложениями. Если вы хотите сделать нечто большее или KDE и GNOME не соответствуют вашим потребностям, можно воспользоваться другими инструментальными средствами, которые стоят отдельного упоминания. В этом разделе коротко рассказывается о некоторых из наиболее популярных инструментальных средств и библиотек мультимедиа, доступных для Linux.

Simple DirectMedia Layer (SDL)

Simple DirectMedia Layer (SDL) – это мультимедийная кроссплатформенная библиотека, основная цель которой состоит в том, чтобы обеспечить низкоуровневый доступ к аудиоустройствам, клавиатуре, мыши, джойстику, трехмерному графическому ускорителю через OpenGL и кадровые видеобуферы. Она используется программным обеспечением воспроизведения формата MPEG, эмуляторами и многими популярными играми, включая заслуживающую награды реализацию игры Civilization для Linux – Call To Power.

Библиотека SDL написана на языке программирования C, но прекрасно интегрируется с программами, написанными на языке C++, и имеет связующие модули для некоторых других языков программирования, включая языки Ada, Eiffel, Java, Lua, ML, Perl, PHP, Pike, Python и Ruby.

Домашняя страница проекта: *http://www.libsdl.org*.

OpenGL

OpenGL – это стандартизированный прикладной программный интерфейс (API) создания двух- и трехмерных графических изображений, разработанный компанией Silicon Graphics, Inc. (SGI). Поддерживает возможность выполнения преобразований, наложения текстур, воспроизведения специальных эффектов и многие другие функции визуализации. Подробную информацию об OpenGL можно найти на сайте *http://www.opengl.org*.

Для Linux имеются несколько свободных реализаций OpenGL. Наиболее популярной является библиотека Mesa. Поскольку она не была лицензирована компанией SGI, официально она не может называться OpenGL, но тем не менее они совместимы. Домашняя страница проекта Mesa: *http://www.mesa3d.org*.

OpenAL

OpenAL – это платформонезависимый прикладной программный интерфейс воспроизведения объемного звука, пригодный для использования в игровых программах и других приложениях, работающих со звуком. Концептуально OpenAL можно представить себе как библиотеку для работы с трехмерным звуком, подобно библиотеке OpenGL, предназначенной для работы с трехмерными изображениями.

Домашняя страница проекта: *http://www.openal.org*.

JACK

JACK – это звуковой сервер, характеризующийся незначительными задержками, разработанный для POSIX-совместимых операционных систем, таких как GNU/Linux и Mac OS X. Позволяет одновременно подключать несколько различных приложений к звуковому устройству, благодаря чему приложения могут совместно использовать одно и то же устройство. Клиенты сервера могут исполняться в виде самостоятельных процессов (то есть как обычные приложения) или работать под управлением самого сервера JACK (т. е. как модули расширения). Домашняя страница проекта JACK: *http://jackit.sourceforge.net*.

GStreamer

GStreamer – это библиотека, которая позволяет конструировать компоненты обработки мультимедиа от простых проигрывателей звуковых файлов до сложнейших аудиомикшеров и программ нелинейного видеомонтажа. Приложения могут использовать усовершенствованные кодеки и выполнять прозрачное фильтрование. Разработчики могут добавлять новые кодеки и фильтры, просто написав модуль расширения, использующий простой и универсальный интерфейс.

Веб-сайт проекта GStreamer: *http://gstreamer.freedesktop.org*.

Network Multimedia Middleware (NMM)

NMM – это пакет промежуточного мультимедийного программного обеспечения, позволяющего создавать распределенные приложения мультимедиа. В состав пакета входит большое число модулей расширения, поддерживающих различные типы мультимедиа, операции и устройства ввода-вывода. Пакет NMM использовался для реализации мультимедийного приложения, представляющего собой расширяемую домашнюю систему развлечений, позволяющую воспроизводить содержимое дисков DVD/CD, выполнять видеозахват, просматривать телепередачи со сдвигом по времени, производить видеозапись, а также создавать списки воспроизведения из любых типов мультимедиа и проигрывать их.

Дополнительные сведения можно найти на сайте *http://www.networkmultimedia.org*.

Media Applications Server (MAS)

Media Applications Server (MAS) – проверенное временем программное обеспечение разрешения конфликтов для видео- и аудиоаппаратуры. Обладает широчайшим диапазоном использования, от встраиваемых систем до систем с массовым параллелизмом, от карманных устройств до суперкомпьютеров, от простейшего микрофона до сложнейших систем распознавания речи. MAS – это сервер маршрутизации мультимедийных потоков. Его главная цель состоит в том, чтобы обеспечить передачу мультимедиа через Интернет практически в режиме реального времени и гарантировать качественную передачу видео-, аудио- и другой информации, чувствительной ко времени передачи.

Дополнительные сведения о MAS можно найти на сайте проекта: *http://www.mediaapplicationserver.net*.

Мультимедийные дистрибутивы

Существуют такие дистрибутивы Linux, которые специально разрабатывались и оптимизировались для работы в качестве мультимедийной платфор-

мы. Один из таких проектов – AGNULA, название которого происходит от «A GNU/Linux Audio». В рамках этого проекта, финансируемого Европейской Комиссией, были выработаны две рекомендации по сборке дистрибутивов Linux из свободного программного обеспечения: DeMuDi (Debian Multimedia Distribution – мультимедийный дистрибутив на базе Debian) и ReHMuDi (Red Hat Multimedia Distribution – мультимедийный дистрибутив на базе Red Hat). Домашняя страница проекта: *http://www.agnula.org*.

Устранение наиболее распространенных неполадок

В данном разделе перечисляются некоторые часто возникающие проблемы, связанные с работой мультимедийной аппаратуры, и способы их решения:

Почему в состав моего дистрибутива не входит кодек файлов формата MP3 или проигрыватель DVD?

Из-за юридических проблем, связанных с патентами, многие дистрибутивы Linux не включают в себя кодек формата MP3 или приложение-проигрыватель DVD. Вы можете загрузить их отдельно после того, как убедитесь, что имеете право использовать их в своей стране.

Существуют ли свободные альтернативы MP3 и DVD, не обремененные патентами?

В рамках проекта Ogg, основанного фондом Xiph.org Foundation, были разработаны несколько форматов кодирования и их свободные реализации, которые свободны от проблем, связанных с патентами. Сюда относятся аудиоформат Ogg Vorbis и видеоформат Ogg Theoris. Дополнительную информацию вы найдете на сайте *http://www.xiph.org*.

Модули ядра не загружены

Это может быть вызвано ошибками в файлах конфигурации модулей или тем, что не запущен загрузчик модулей ядра (*kerneld* или *kmod*). Проверьте наличие загружаемого модуля в нужном каталоге (обычно вида */lib/ modules/2.4.17/kernel/drivers/sound*).

Звуковая карта не обнаруживается

Возможно, используется не тот модуль ядра, который нужен, либо неверно заданы значения адреса I/O, IRQ или канала DMA.

Тайм-аут IRQ/DMA или конфликты устройств

Указаны неверные значения адреса I/O, IRQ и DMA либо существует конфликт с другой картой, использующей те же настройки.

После перезагрузки звук отсутствует

Если звук был, но исчез после перезагрузки системы, это может указывать на проблему с файлами конфигурации модулей. Это может произойти и тогда, когда системные сценарии *init* не настроены на инициализацию карт PnP или загрузку модулей. Если драйверы загружены, то, возможно, в микшере установлена слишком низкая громкость. Попробуйте с помощью программы-микшера настроить уровни громкости во время проигрывания музыкального файла.

Если вы пользуетесь рабочим столом KDE или GNOME, убедитесь, что был запущен соответствующий звуковой сервер (*aRts* или *esd*). Некоторые серверы требуют выполнения дополнительной настройки с помощью панели управления, предусмотренной специально для этих целей. В случае KDE такой панелью является приложение Центр управления.

Звук воспроизводится только для root

Это может указывать на неправильную установку прав доступа к файлам устройств. Во многих системах доступ к звуковым устройствам разрешен только пользователям, входящим в группу audio. Добавьте в эту группу пользователей или измените права доступа к звуковым устройствам с помощью команды `chmod`. Некоторые версии ядра Linux из ветки 2.6 игнорируют настройки прав доступа группы для файлов устройств, поэтому необходимо сделать их доступными для всех пользователей в системе.

Звука не слышно, но сообщений об ошибках нет

Если программы звуковоспроизведения по виду работают, но звука не слышно, это может быть связано с настройками микшера или надежностью подключения звуковых колонок.

Невозможно записать звук

Это может указывать на проблему с настройками микшера. Необходимо установить уровни громкости и выбрать входное устройство. Возможно также, что у вас неисправен микрофон или вы включили его в неверное гнездо на звуковой карте.

Ошибка: устройство занято

Либо у вас конфликт устройств, либо другое приложение использует звуковые устройства. Это может быть связано с работой программы – сервера звука, такой как *esd* или *artsd*.

Отсутствие звука при воспроизведении звукового компакт-диска

Для воспроизведения звуковых компакт-дисков необходимо соединить кабелем привод CD-ROM и звуковую карту. Проверьте, выбрали ли вы вход CD в программе-микшере. Попробуйте включить наушники в гнездо на передней панели привода CD-ROM; если звук слышен, следовательно, проблема не в самом приводе. Если в наушниках звука нет, значит, проблема связана с приводом или программой воспроизведения CD.

Невозможно воспроизведение MIDI-файлов

Некоторые приложения MIDI работают только с картами на основе ЧМ-синтезатора, которого может не быть на вашей карте (либо он не поддерживается драйвером звуковой карты). Другие приложения MIDI используют стандартное аудиоустройство.

Ссылки

Ниже перечислены некоторые информационные ресурсы, имеющие отношение к мультимедиа в Linux.

Sound and MIDI Software For Linux – каталог мультимедийных приложений и ресурсов.

http://sound.condorow.net

SourceForge – крупнейший в мире веб-сайт, посвященный разработке программного обеспечения с открытыми исходными текстами.

http://www.sourceforge.net

Freshmeat – гигантский каталог коммерческого и свободного программного обеспечения.

http://freshmeat.net

Документ «Linux Sound HOWTO», доступен на сайте проекта документирования Linux (LDP).

http://www.tlpd.org

Документ «Linux CD-ROM HOWTO», доступен на сайте проекта документирования Linux (LDP).

http://www.tlpd.org

Проект ALSA.

http://www.alsa-project.org

4Front Technologies.

http://www.opensound.com

Проект KDE.

http://www.kde.org

Проект GNOME.

http://www.gnome.org

Проект WINE.

http://www.winehq.com

CodeWeavers, компания-разработчик CrossOver.

http://www.codeweavers.com

Проект ReWind.

http://rewind.sourceforge.net

TransGaming Technologies, компания-разработчик Cedega.

http://www.transgaming.com

Книга «Linux Multimedia Guide» (изд. O'Reilly).

http://www.oreilly.com/catalog/multilinux/

Книга «Linux Music and Sound».[1]

http://www.nostarch.com/lms.htm

---

[1] Дейв Филлипс «Музыка и звук в Linux. Настольная книга музыканта». – Пер. с англ. – ТИД «ДС», 2005.

II

Системное администрирование

В этой части книги мы покажем, как настроить свою систему Linux и программное окружение, чтобы иметь возможность решать далее более важные задачи, такие как печать и использование файлов совместно с другими системами. Здесь также описываются некоторые из способов обслуживания операционной системы. Если вашей системой пользуется еще кто-то, материал этого раздела будет для вас особенно важен. Этот раздел будет не менее важен, если ваш дистрибутив окажется не в состоянии работать в сети или если вы решите запустить какой-либо из серверов, описываемых в четвертой части книги.

10

Основы системного администрирования

При запуске собственной Linux-системы вскоре становится необходимостью изучение способов системного администрирования. Вам не удастся долго обходиться без осуществления некоторых видов системного сопровождения, обновления программного обеспечения и просто мелочей, необходимых для поддержания дел в порядке.

Работа в Linux-системе не лишена сходства с ездой на мотоцикле и уходом за ним.[1] Многие любители мотоциклетной езды предпочитают сами заботиться о своей технике: регулярно чистить, менять изношенные детали и т. д. Linux дает возможность осуществлять такого же рода практический уход за сложной операционной системой.

Хотя одержимый администратор может бесконечно настраивать систему, чтобы улучшить ее работу, на практике заниматься администрированием необходимо только в случае значительных событий: установки нового диска, появления нового пользователя или выхода системы из строя при неожиданном отказе питания. Все эти ситуации мы будем обсуждать на протяжении следующих четырех глав.

Linux удивительно доступна во всех отношениях – от будничных задач обновления совместно используемых библиотек до таких, понятных лишь посвященным, задач, как копание в ядре. Ввиду доступности всего кода – а основная масса разработчиков и пользователей Linux всегда была из хакерской породы – системное сопровождение не только является частью повседневной работы, но и имеет большое познавательное значение. Поверьте нам: ничто не может сравниться с рассказом друзьям о том, как вам удалось перейти с PHP 4.3 на PHP 5.0 менее чем за полчаса, и при этом вы сумели пересобрать ядро, включив в него поддержку файловой системы ISO 9660. (Они могут понятия не иметь, о чем вы им говорите; в таком случае подарите им экземпляр этой книги.)

В нескольких последующих главах мы будем исследовать систему Linux с точки зрения механика: покажем, что находится под капотом, и объясним, как за всем

---

[1] По крайней мере, один из авторов свидетельствует о сильном сходстве, которое имеют между собой системное администрирование Linux и книга Роберта Пирсинга «Дзен-буддизм и искусство ухода за мотоциклом». Присуща ли Linux многоликость Будды?

этим ухаживать, включая обновление программного обеспечения, управление пользователями, файловыми системами и другими ресурсами, изготовление резервных копий и действия в экстренных ситуациях.

Если вы сделали нужные изменения в стартовых файлах, Linux по большей части будет работать самостоятельно. Пока вас устраивает системная конфигурация и программы, работающие в системе, особых трудозатрат не требуется. Однако мы хотим побудить пользователей Linux к экспериментам со своей системой и настройке ее согласно своим вкусам. Твердокаменность несвойственна Linux, и если что-либо работает не так, как вам того хотелось бы, наверняка есть возможность это изменить. Например, в предыдущих главах мы показывали, как вместо обычного белого текста на черном фоне получить мигающий зеленый текст на голубом фоне или как добавить апплет на панель рабочего стола. Однако в этой книге мы продемонстрируем вам гораздо более важные вещи: как правило, сразу после установки Linux в системе исполняется масса различных служб, в которых порой нет никакой необходимости (например, веб-сервер). Любая из этих служб может представлять угрозу безопасности, поэтому может потребоваться поковыряться в стартовых файлах, чтобы оставить работать только те службы, которые действительно необходимы.

Следует отметить, что в состав Linux часто включаются замысловатые инструментальные средства, упрощающие решение многих задач системного администрирования. В число таких средств входят YaST в системах SUSE, Mandriva Control Center (центр управления Mandriva) в системах Mandriva и ряд утилит в системах Red Hat. С их помощью можно делать все – от управления учетными записями пользователей до создания файловых систем и сборки мусора. Эти утилиты могут как облегчить, так и затруднить вашу жизнь в зависимости от того, как на них взглянуть. В этой и последующих главах мы представим «нутро» системного администрирования, показав средства, которые должны существовать в любой системе Linux и в действительности почти в любой UNIX-системе. Они составляют основу инструментального ящика администратора: его молоток, отвертку и торцевой ключ, благодаря которым можно справиться с поставленной задачей. Если вы хотите воспользоваться 40-сильной дисковой пилой, это ваше право, но всегда полезно уметь пользоваться ручным инструментом на случай отключения электроэнергии. При желании подробнее изучить другие темы по администрированию UNIX мы рекомендуем книги Эви Немет (Evi Nemeth) и др. «UNIX System Administration Handbook»[1] и Элин Фриш (Жleen Frisch) «Essential System Administration» (O'Reilly).

Сопровождение системы

Работа системного администратора любой UNIX-системы в определенной мере требует чувства ответственности и осторожности. В равной степени это применимо к Linux, даже если вы единственный пользователь системы.

Многие из задач системного администрирования выполняются пользователем *root*. В UNIX-системах эта учетная запись обладает особыми правами: обычные

[1] Э. Немет, Г. Снайдер, С. Сибасс, Т. Хейн «UNIX: руководство системного администратора. Для профессионалов», 3-е издание. – Пер. с англ. – СПб.: Питер, 2007.

права доступа к файлам и механизмы системы безопасности просто не применяются к *root*. Таким образом, *root* может обратиться к любому файлу системы вне зависимости от его принадлежности и модифицировать его. В то время как обычный пользователь не может повредить систему (например, путем повреждения файловой системы или действий с файлами других пользователей), *root* таких ограничений не имеет.

Здесь следует упомянуть, что в некоторых дистрибутивах, таких как Ubuntu, доступ к учетной записи пользователя *root* закрыт, а все действия, требующие привилегий суперпользователя, в этих дистрибутивах производятся с помощью утилиты *sudo*. Утилита *sudo* не дает возможности зарегистрироваться в системе с учетной записью *root*, но она позволит выполнить точно одну команду с правами *root*. Это по сути ничего не меняет, за исключением того, что вам придется предварять каждую команду вызовом *sudo*.[1]

Почему в UNIX защита стоит на первом месте? Наиболее очевидная причина в том, чтобы разрешить пользователям самим определять порядок доступа к своим файлам. Изменяя биты разрешения на доступ к файлу (командой *chmod*), пользователи могут указать, что некоторые файлы доступны только для чтения, записи или выполнения определенным группам других пользователей или недоступны вообще никому. Права доступа позволяют обеспечить конфиденциальность и целостность данных; например, вам бы не хотелось, чтобы другие пользователи могли читать вашу личную почту или редактировать втайне от вас исходный код важной программы.

Механизмы защиты UNIX предотвращают также повреждение системы пользователями. Система ограничивает прямой доступ ко многим файлам устройств (доступным через */dev*; подробнее об этом мы поговорим в разделе «Файлы устройств» далее в этой главе), связанным с аппаратными устройствами, такими как жесткие диски. Если бы обычные пользователи имели возможность чтения и записи непосредственно на дисковые устройства, это могло бы привести к большим неприятностям, когда, например, содержимое диска оказалось бы полностью затертым. Вместо этого система требует, чтобы обычные пользователи обращались к диску через файловую систему, которая обеспечивает систему защиты через биты прав доступа, описанных ранее.

Важно отметить, что не все виды ущерба, который может быть нанесен, обязательно являются зловредными. Система безопасности скорее призвана защитить пользователей от собственных случайных ошибок и недосмотров, а не устанавливать полицейские порядки. И действительно, во многих операционных системах подсистема защиты довольно слаба. Система безопасности UNIX создана с целью обеспечить возможность совместного доступа к данным группе пользователей, которые могут, например, совместно работать над некоторым проектом. Система позволяет включать пользователей в группы, а права доступа к файлу могут устанавливаться для группы в целом. Например, в некотором разрабатываемом проекте у одних пользователей могут быть права чтения и записи для ряда файлов, а другим пользователям может быть запрещено модифицировать эти фай-

[1] Подобного эффекта можно добиться в большинстве UNIX-систем (за исключением FreeBSD, где это запрещено), введя команду *su* с терминала; если вам известен пароль *ru*, вы продолжите дальнейший сеанс от имени *root*. – *Примеч. науч. ред.*

лы. В отношении личных файлов вы сами должны решить, какую степень открытости или закрытости должны обеспечивать разрешения.

Механизм системы безопасности UNIX не разрешает также обычным пользователям производить некоторые действия, например обращаться в программе к определенным системным вызовам. Например, существует системный вызов, выполняющий останов системы и используемый такими программами, как *shutdown* (подробнее об этом ниже), для перезагрузки системы. Если бы обычным пользователям было разрешено обращаться к этой функции из своих программ, они могли бы в любой момент случайно (или преднамеренно) остановить систему.

Во многих случаях требуется обойти механизмы защиты UNIX, чтобы выполнить операции по сопровождению системы или обновлению программного обеспечения. Для этого служит учетная запись *root*. Поскольку на нее не накладываются никакие ограничения, знающий системный администратор может без труда выполнить задачу, не заботясь об обычных правах доступа к файлам или других ограничениях. Обычно для регистрации в качестве *root* используется команда *su*. Эта команда позволяет действовать от лица другого пользователя, например

```
su andy
```

выведет приглашение для ввода пароля пользователя *andy*, и если пароль верный, установит ID (числовой идентификатор) пользователя *andy*. Суперпользователю часто приходится на время регистрироваться под учетными записями других пользователей, чтобы исправить проблемы с файлами этого пользователя или по каким-то аналогичным причинам. Если не указывать имя пользователя, то *su* потребует ввести пароль пользователя *root*, проверяя, являетесь ли вы пользователем *root*. Закончив пользоваться учетной записью *root*, вы обычным способом завершаете сеанс работы и возвращаетесь к собственному смертному обличью.[1]

Почему просто не зарегистрироваться как *root* через обычное приглашение регистрации? Как мы увидим, в некоторых случаях это желательно, но, как правило, лучше зарегистрироваться под своим именем, а затем воспользоваться командой *su*. В системе со многими пользователями обращение к *su* создает в системных журналах, таких как */var/log/messages* (о файлах системных журналов мы поговорим позднее), запись типа:

```
Nov 1 19:28:50 loomer su: mdw on /dev/ttyp1
```

Это сообщение говорит о том, что пользователь *mdw* успешно выполнил команду *su*, получив в данном случае привилегии пользователя *root*. Если бы вы сразу регистрировались в системе с привилегиями суперпользователя, никаких сообщений в системном журнале не было бы, и какой пользователь зарегистрировался как *root*, сказать было бы невозможно. Это важно, если на машине несколько администраторов: часто желательно знать, кто и когда пользовался командой *su*.

---

[1] Обратите внимание: ядро UNIX совершенно не интересует имя пользователя, то есть *root*: оно будет считать суперпользователем любого, чей идентификатор имеет значение 0. По умолчанию имя *root* – это единственное имя пользователя в системе, которое имеет значение идентификатора 0, но вы вправе, например, создать пользователя *thebigboss* и также присвоить ему идентификатор со значением 0. Как это сделать, мы расскажем в следующей главе.

Есть еще один небольшой фокус, связанный с командой *su*. Выполнив ее так, как описано выше, вы измените свой ID пользователя, но при этом не получите все настройки, связанные с этим ID. У каждого пользователя могут быть особые файлы настройки (об их создании еще будет рассказано), но при таком использовании *su* они не будут выполняться. Чтобы эмулировать реальную регистрацию с выполнением всех файлов настроек, нужно добавить символ -:

```
su - andy
```

или

```
su -
```

чтобы стать пользователем *root* и получить окружение пользователя *root*.

Учетную запись *root* можно считать волшебной палочкой – полезным и потенциально опасным средством. Неумелое обращение с волшебными словами, пока у вас в руках эта палочка, может принести несказанный вред системе. Например, всего лишь восемь символов `rm -rf /` уничтожат все файлы в системе, если вы зарегистрированы как *root* и невнимательны. Эта проблема кажется искусственной? Отнюдь нет. Вы могли попытаться удалить старый каталог, такой как */usr/src/oldp*, и случайно вставили пробел после первого символа слэша:

```
rm -rf / usr/src/oldp
```

Источником проблем могут оказаться и каталоги, имена которых содержат пробелы. Допустим, что имеются каталоги *Dir\ 1* и *Dir\ 2*, где символ обратного слэша означает, что *Dir\ 1* – это единое имя файла, в котором есть пробел. Представим, что понадобилось удалить оба каталога, но по ошибке снова был вставлен лишний пробел:

```
rm 'rf Dir\ *
```

Между обратным слэшем и звездочкой оказалось два пробела. Первый из них «экранируется» обратным слэшем, а второй – нет, поэтому он разделяет аргументы и делает звездочку новым аргументом. Увы, текущий каталог и все, что в нем находится, будут уничтожены.

Другой частой ошибкой является путаница в аргументах команд, таких как `dd`, которая часто используется для перемещения больших объемов данных из одного места в другое. Например, чтобы сохранить первые 1024 байта данных с устройства */dev/hda* (в которых содержатся загрузочная запись и таблица разделов этого диска), можно выполнить команду:

```
dd if=/dev/hda of=/tmp/stuff bs=1k count=1
```

Однако если в этой команде поменять местами `if` и `of`, то произойдет совершенно иное: содержимое файла */tmp/stuff* будет записано поверх */dev/hda*. Скорее всего, вам только что удалось убить свою таблицу разделов и, возможно, суперблок файловой системы.[1] Добро пожаловать в чудесный мир системного администрирования!

---

[1] То есть не только свою установленную операционную систему Linux, но и несколько других операционных систем (Windows, FreeBSD), установленных на тот же диск. – *Примеч. науч. ред.*

Все это сказано к тому, что, прежде чем выполнять какую-либо команду с привилегиями пользователя *root*, посидите некоторое время, ничего не делая. Посмотрите внимательно на команду в течение минуты, прежде чем давить на Enter, и убедитесь, что она верная. Если вы не уверены в аргументах и синтаксисе команды, быстренько посмотрите страницы руководства или проверьте ее в безопасном окружении, прежде чем запускать на выполнение. Иначе вы можете получить суровый урок. Ошибки, совершенные от имени *root*, могут быть катастрофическими.

Полезный совет: можно воспользоваться командой *alias*, чтобы сделать некоторые команды, выполняемые с привилегиями суперпользователя, менее опасными. Например, можно выполнить команду:

```
alias rm="rm -i"
```

Параметр *-i* происходит от слова «interactively» («интерактивно») и означает, что команда *rm* станет запрашивать подтверждение перед удалением каждого файла. Правда, это не защитит вас от жестокой ошибки, показанной выше: параметр *-f* (от слова «force» – принудительно) просто отменит действие параметра *-i*, поскольку будет стоять в командной строке вслед за ним.

Часто приглашение для учетной записи *root* выглядит не так, как для обычных пользователей. В классическом случае приглашение к вводу для пользователя *root* содержит символ #, тогда как для обычных пользователей это $ или %. (Разумеется, вам решать, следовать ли этому соглашению, однако оно используется во многих UNIX-системах.) Хотя внешний вид приглашения к вводу может напомнить о том, что у вас в руках волшебная палочка *root*, тем не менее пользователи часто забывают об этом или по ошибке вводят команду не в том окне или не в той виртуальной консоли.

Как и любым мощным инструментом, учетной записью *root* можно злоупотребить. Системному администратору необходимо хранить в секрете пароль пользователя *root*, и если вы вообще кому-нибудь его раскрываете, делайте это исключение только для тех пользователей, которым вы доверяете (или с которых можно спросить за их действия в системе). Если вы единственный пользователь системы, то эта информация к вам не относится, если только вы не подключены к сети или не разрешаете доступ к системе по коммутируемому соединению.

Основное преимущество единоличного использования учетной записи *root* состоит не в уменьшении возможностей злоупотребления, хотя и это важно. Еще более важно, что если вы один можете использовать учетную запись *root*, то вы и имеете полное знание о том, как настроена система. Если кто-либо еще обладает правами, предоставляющими возможность изменять важные системные файлы (о чем мы расскажем в этой главе), то настройки системы могут быть изменены без вашего ведома, и ваши предположения о том, как работает система, оказываются неверными. Наличие единственного администратора, выступающего в качестве арбитра настраиваемых параметров системы, означает, что есть человек, который всегда знает, что происходит на самом деле.

Кроме того, если пароль *root* знает кто-то еще, возникает возможность того, что, в конце концов, он сделает ошибку при использовании учетной записи *root*. Даже если всем, кто знает пароль суперпользователя, можно доверять, никто не застрахован от ошибок. Если вы – единственный администратор системы, то остается винить только себя за неизбежные ошибки при использовании учетной записи *root*.

После всего сказанного займемся реальными задачами администрирования под Linux. Пристегните ремни.

Управление файловыми системами

Вероятно, вам уже пришлось столкнуться с созданием файловых систем и раздела подкачки во время установки Linux (большинство дистрибутивов окажут вам в этом посильную помощь). Теперь настало время выполнить тонкую настройку этих ресурсов. В большинстве случаев эти настройки выполняются сразу же после установки операционной системы, прежде чем вы начнете заполнять диски всякими полезными вещами. Но иногда возникает необходимость выполнить эти действия на уже работающей системе, например, при добавлении нового устройства или для изменения размера раздела подкачки после увеличения объема оперативной памяти.

В UNIX-системах файловая система представляет собой некоторое устройство (например, жесткий диск, гибкий диск или компакт-диск), отформатированное для хранения файлов. Файловые системы могут находиться на жестких дисках, гибких дисках, CD-ROM или других носителях, которые позволяют осуществлять произвольный доступ. (Замечание: магнитная лента позволяет осуществлять только последовательный доступ и потому не может содержать файловую систему как таковую.)

Точный формат и способы хранения файлов не имеют значения; система обеспечивает общий интерфейс для всех распознаваемых ею *типов файловых систем*. В Linux в число файловых систем входит «третья расширенная файловая система», или *ext3fs*, в которой вы, возможно, уже храните файлы Linux, файловая система VFAT, позволяющая из Linux иметь доступ к файлам на разделах Windows 95/98/ME и гибких дисках (а также на разделах Windows NT/2000/XP, если они отформатированы как файловые системы FAT), а также некоторые другие, в том числе файловая система ISO 9660, используемая на CD-ROM.

Все эти файловые системы имеют различные внутренние форматы хранения данных. Однако при доступе к любой файловой системе из Linux данные представляются в виде иерархии каталогов с находящимися в них файлами вместе с идентификаторами владельцев и групп, битами прав доступа и прочими, уже знакомыми характеристиками.

Фактически данные о владельцах, правах доступа и прочее обеспечиваются только файловыми системами, предназначенными для хранения файлов Linux. Для файловых систем тех типов, которые не хранят такие данные, драйверы ядра, используемые для доступа к ним, «подделывают» эти сведения. Например, в файловой системе MS-DOS нет понятия владельца файла, поэтому все файлы представляются так, как если бы их владельцем был *root*. Благодаря этому начиная с некоторого уровня все файловые системы выглядят примерно одинаково, и у каждого файла есть некоторые связанные с ним атрибуты. Действительно ли эти данные используются внутри файловой системы – это уже другой вопрос.

Как системный администратор вы должны уметь создавать файловые системы, когда вам требуется сохранить файлы Linux на гибком диске или добавить новые файловые системы на жестких дисках. Вы должны также знать, как пользоваться различными средствами для проверки и поддержки файловых систем, ес-

ли данные окажутся повреждены. Кроме того, нужно знать команды и файлы, используемые для доступа к файловым системам, например на гибком диске или CD-ROM.

Типы файловых систем

В табл. 10.1 перечислены типы файловых систем, поддерживаемых ядром Linux версии 2.6.5. К системе постоянно добавляются новые файловые системы, и для некоторых файловых систем есть экспериментальные драйверы, здесь не отмеченные. Чтобы узнать, какие типы файловых систем поддерживает ваше ядро, взгляните на файл */proc/filesystems*. Файловые системы, которые необходимо поддерживать, можно выбрать при сборке ядра (см. раздел «Настройка ядра: make config» в главе 18).

Таблица 10.1. Типы файловых систем, поддерживаемых Linux

Файловая система	Тип	Описание
Second Extended	*ext2*	Наиболее широко используемая в Linux файловая система. Постепенно вытесняется более современными файловыми системами Reiser и Third Extended
Reiser	*reiserfs*	Журналируемая файловая система для Linux
Third Extended	*ext3*	Еще одна журналируемая файловая система для Linux, обладающая обратной совместимостью с файловой системой *ext2*
JFS	*jfs*	Журналируемая файловая система для Linux, реализованная компанией IBM. Используется как альтернатива файловым системам *reiserfs* и *ext3*
Network File System (NFS)	*NFS*	Позволяет организовать сетевой доступ к удаленным файлам
UMSDOS	*umsdos*	Используется при установке Linux на дисковые разделы MS-DOS
DOS-FAT	*msdos*	Используется для доступа к файлам операционной системы MS-DOS
VFAT	*vfat*	Используется для доступа к файлам семейства операционных систем Windows 95/98/ME
NT	*ntfs*	Используется для доступа к файлам семейства операционных систем Windows NT/2000/XP
/proc	*proc*	Используется для хранения информации о запущенных процессах
ISO 9660	*iso9660*	Используется большинством CD-ROM
UDF	*udf*	Более современная файловая система для CD-ROM
SMB	*smbfs*	Позволяет организовать сетевой доступ к файлам на серверах, работающих под управлением операционной системы Windows

Файловая система	Тип	Описание
Coda	*coda*	Улучшенная сетевая файловая система, напоминающая NFS
Cifs	*cifs*	Common Internet File System (универсальная файловая система Интернета) была предложена корпорацией Microsoft в качестве замены SMB. Поддерживается операционными системами Windows 2000, 2003 и XP, так же как и сервер Samba

Файловая система каждого типа имеет свои атрибуты и ограничения. Например, файловая система MS-DOS ограничивает длину имен файлов восемью символами и тремя символами расширения и должна использоваться только для доступа к имеющимся гибким дискам и разделам MS-DOS. При работе с Linux вы в основном будете использовать вторую или третью расширенные файловые системы, разработанные в первую очередь для Linux. Они поддерживают имена файлов длиной до 256 символов, позволяют создавать разделы размером до 32 терабайт и имеют массу прочих достоинств. Можно также воспользоваться файловой системой Reiser (*reiserfs*). Ранее в Linux использовались файловые системы Extended (более не поддерживается) и Minix. (Первоначально по ряду причин использовалась файловая система Minix. Во-первых, Linux изначально кросс-компилировалась под Minix. Кроме того, Линус хорошо знал файловую систему Minix, и ее было просто реализовать в первых вариантах ядра.) Некоторые другие файловые системы, имевшие реализации в старых версиях ядра Linux, больше не поддерживаются.

Основное отличие второй расширенной файловой системы от файловой системы Reiser и третьей расширенной файловой системы в том, что последние две являются журналируемыми. Журналирование – это передовая технология, при использовании которой отслеживаются изменения, выполненные в файловой системе, что значительно облегчает (и ускоряет!) восстановление поврежденных файловых систем (например, после краха системы или отключения питания). Еще одна журналируемая файловая система – Journaling File System (JFS), реализованная компанией IBM.

Файловая система ROM едва ли вам понадобится. Она очень маленькая, не поддерживает операции записи и предназначена для организации электронных дисков в оперативной памяти (и даже EPROMS) при настройке системы во время запуска. К той же категории относится файловая система Cram, которая также используется для работы в ПЗУ и сжимает свои данные. Она предназначена в основном для встроенных устройств, где ценится экономия памяти.

Файловая система UMSDOS используется для установки Linux в закрытый каталог на существующем разделе MS-DOS. Это хороший способ для новичков потренироваться с Linux, не производя разбиения диска на разделы. Напротив, файловая система DOS-FAT используется для прямого доступа к файлам MS-DOS. Доступ к файлам разделов, созданных с помощью Windows 95 или 98, может быть осуществлен с помощью файловой системы VFAT, а файловая система NTFS осуществляет доступ к файловым системам Windows NT. Файловая система HPFS используется для доступа к файловой системе OS/2.

Файловая система */proc* является виртуальной файловой системой, то есть она реально не занимает дисковое пространство. Более подробные сведения о файловой системе */proc* вы найдете в разделе «Файловая система /proc» далее в этой главе.[1]

Файловая система ISO 9660 (ранее известная как High Sierra Filesystem, сокращенно – *hsfs*) используется на большинстве CD-ROM. Подобно MS-DOS, эта файловая система ограничивает длину имени файла и хранит лишь часть информации о каждом файле. Однако в большинстве CD-ROM применяется расширение Rock Ridge Extensions для ISO 9660, позволяющее драйверу файловой системы ядра присваивать каждому файлу длинные имена, права владения и доступа. В результате при доступе к ISO 9660 CD-ROM из MS-DOS получаются имена в формате 8.3, но при доступе из Linux получаются «подлинные» полные имена файлов.

Кроме того, Linux теперь поддерживает расширение Microsoft Joliet для ISO 9660, позволяющее обращаться с длинными именами в кодировке Unicode. В настоящее время эта кодировка используется не так широко, но в будущем может стать ценным достоинством, поскольку кодировка Unicode стала общепринятым стандартом кодировки символов.

Недавно в Linux была также включена поддержка UDF – файловой системы, предназначенной для использования с CD-RW и DVD.

Кроме вышеперечисленных, ядро Linux обеспечивает поддержку еще целого ряда других файловых систем, используемых на других платформах, что позволяет иметь на одном диске две операционные системы и осуществлять другие виды взаимодействия. Речь идет о файловых системах UFS, EFS, BFS, XFS, System V и BeOS. Если у вас есть файловые системы, созданные в одном из этих форматов в другой операционной системе, вы сможете получить к ним доступ из Linux.

Наконец, есть масса файловых систем для доступа к данным на разделах, созданных в операционных системах, принадлежащих иным семействам, нежели DOS и UNIX. Из них поддерживаются файловые системы: Acorn Disk Filing System (ADFS), AmigaOS (без поддержки гибких дисков, за исключением компьютеров Amiga), Apple Mac HFS и QNX4. Большинство из этих файловых систем имеет смысл использовать только на отдельных аппаратных архитектурах. Например, на машине Intel вы не найдете жестких дисков, отформатированных в файловой системе Amiga FFS. Если вам понадобится один из этих драйверов, прочтите информацию, поставляемую вместе с ним. Некоторые из них находятся в экспериментальной стадии развития.

Помимо файловых систем, предназначенных для доступа к локальным жестким дискам, существуют сетевые файловые системы, позволяющие организовать доступ к удаленным ресурсам. Мы поговорим о них немного погодя.

И в довершение существуют специализированные файловые системы, которые используются для организации хранения данных в оперативной памяти (по этой

---

[1] Учтите, что файловая система */proc* имеет другой формат, нежели файловая система */proc* в SVR4 (как, например, в Solaris 2.x). В SVR4 каждый выполняющийся процесс имеет в */proc* один «файл», который можно открыть и производить с ним действия с помощью некоторых вызовов *ioctl( )*, чтобы получить сведения о процессе. Напротив, Linux предоставляет большинство сведений в */proc* с помощью запросов *read( )* и *write( )*.

причине они отличаются чрезвычайно высоким быстродействием, но данные в этих файловых системах безвозвратно теряются с выключением питания) и предоставления доступа к объектам и данным ядра.

Монтирование файловых систем

Для доступа к любой файловой системе из Linux ее нужно смонтировать в некотором каталоге, в результате чего ее файлы будут выглядеть так, будто они находятся в данном каталоге и доступны вам.

Прежде чем рассказывать о том, как монтируются файловые системы, отметим, что в некоторых дистрибутивах изначально настроена функция автоматического монтирования, в этом случае достаточно лишь вставить дискету или CD в соответствующий привод, после чего можно работать с ними, как на других платформах. Однако бывают случаи, когда нужно уметь смонтировать и размонтировать носитель вручную.

Операция монтирования обычно осуществляется с использованием команды *mount*, для запуска которой обычно требуются привилегии суперпользователя. (Как мы увидим ниже, обычные пользователи также могут выполнять команду *mount*, если устройство перечислено в файле */etc/fstab*.) Команда имеет следующий формат:

```
mount -t type device mount-point
```

где аргумент *type* представляет тип файловой системы в соответствии с табл. 10.1, *device* – физическое устройство, на котором находится файловая система (файл устройства в файловой системе */dev*), а *mount-point* – имя каталога, к которому должна быть примонтирована файловая система. Этот каталог должен быть создан прежде[1], чем будет запущена команда *mount*.

Например, если на разделе */dev/hda2* находится третья расширенная файловая система и вы хотите смонтировать ее в каталог */mnt*, выполните команду:

```
mount -t ext3 /dev/hda2 /mnt
```

При благоприятном исходе вы сможете обращаться к файловой системе в каталоге */mnt*. Аналогично для монтирования гибкого диска, созданного в Windows и потому имеющего формат DOS, нужно использовать команду:

```
mount -t msdos /dev/fd0 /mnt
```

В результате файлы, имеющиеся на гибком диске в формате MS-DOS, будут доступны в каталоге */mnt*. Обратите внимание, что аргумент `msdos` означает использование старого формата DOS, допускающего для имен файлов 8 символов плюс 3 символа расширения. Если задать вместо него `vfat`, вы получите более новый формат, введенный в Windows 95. Конечно, гибкий или жесткий диск также должен быть записан в этом формате.

У команды *mount* имеется множество параметров, которые можно задавать с ключом `-o`. Например, файловые системы MS-DOS и ISO 9660 поддерживают

[1] Это требование не является общим во всех UNIX-системах, но в Linux это именно так. – *Примеч. науч. ред.*

«автоконвертирование» текстовых файлов из формата MS-DOS (они содержат в конце каждой строки пару символов CR-LF) в формат UNIX (в котором в конце каждой строки есть только символ перевода строки – LF). Например, такая команда, как

```
mount -o conv=auto -t msdos /dev/fd0 /mnt
```

включает это преобразование для файлов, расширения которых не связываются с двоичными файлами (*.exe*, *.bin* и т. д.).

При монтировании часто используется параметр `-o ro` (эквивалентно `-r`), в результате чего файловая система монтируется только для чтения. На все попытки записи в такую файловую систему будет возвращаться сообщение об ошибке: «permission denied». Монтирование файловой системы в режиме только для чтения необходимо для таких носителей, как CD-ROM, на который нельзя осуществлять запись. Можно благополучно смонтировать CD-ROM и без параметра `-r`, но при этом вы получите раздражающее предупредительное сообщение:

```
mount: block device /dev/cdrom is write-protected, mounting read-only
```

Лучше выполните команду

```
mount -t iso9660 -r /dev/cdrom /mnt
```

Это также необходимо и при монтировании гибкого диска с установленной на нем защитой от записи.

На страницах справочного руководства по команде `mount` перечислены все имеющиеся параметры. Не все из них представляют непосредственный интерес, но когда-нибудь они вам, возможно, понадобятся. Полезен вариант команды *mount -a*, который монтирует все файловые системы, перечисленные в */etc/fstab*, исключая помеченные флагом `noauto`.

Обратной операцией по отношению к монтированию является, естественно, размонтирование. Размонтирование файловой системы имеет два последствия: происходит синхронизация системных буферов с фактическим содержанием файловой системы на диске, и файловая система перестает быть доступной через свою точку монтирования (mount point). После размонтирования вы можете смонтировать в той же точке другую файловую систему.

Размонтирование производится командой *umount* (обратите внимание на отсутствие первого «n» в слове «unmount»), например:

```
umount /dev/fd0
```

Эта команда размонтирует файловую систему на */dev/fd0*. Аналогично для размонтирования любой файловой системы, смонтированной в данное время в некотором каталоге, используйте команду

```
umount /mnt
```

Важно отметить, что сменные носители, в том числе дискеты и CD-ROM, нельзя вынимать из дисковода или заменять другими, пока смонтирована файловая система. В результате такого действия возникает рассинхронизация системной информации об устройстве с его фактическим состоянием, что может привести к бесконечным проблемам. Если вы хотите поменять дискету или компакт-диск,

сначала отмонтируйте их командой *umount*, а затем вновь смонтируйте. Конечно, если речь идет о CD-ROM или защищенной от записи дискете, то само устройство не может рассинхронизироваться, но могут возникнуть другие проблемы. Например, вы не сможете вынуть компакт-диск из привода, пока он не будет размонтирован.

Чтение и запись на гибкие диски буферизуются в памяти так же, как для жестких дисков. Это значит, что при чтении или записи можно не увидеть сразу работу дисковода. Система обрабатывает ввод-вывод на дискеты асинхронно и читает или записывает данные только тогда, когда это абсолютно необходимо. Поэтому, если вы записываете на дискету маленький файл и лампочка привода при этом не загорается, не впадайте в панику – в свое время данные будут записаны. Принудить систему записать на диск все буферы файловой системы можно командой `sync`, которая осуществит фактическую запись буферизованных данных на физический носитель. То же самое происходит при размонтировании файловой системы.

Если вы хотите, чтобы и обычные пользователи могли производить монтирование и размонтирование некоторых устройств, есть два выхода. Во-первых, можно включить параметр `user` для этого устройства в файл */etc/fstab* (описывается ниже в этом разделе). Это позволит любому пользователю запускать команды `mount` и `umount` для данного устройства. Другой способ – использование одного из интерфейсов монтирования более высокого уровня, имеющихся в Linux. Эти программы выполняются с установленным битом *setuid* и позволяют обычным пользователям монтировать некоторые устройства. Скорее всего, вы не позволите обычным пользователям монтировать и размонтировать разделы на жестких дисках, но к использованию CD-ROM и гибких дисков можно относиться лояльнее.

При попытке монтирования файловой системы может возникнуть немало проблем. К сожалению, команда `mount` выдает одно и то же сообщение об ошибке при возникновении разных проблем:

```
mount: wrong fs type, /dev/cdrom already mounted, /mnt busy или other error
```

`wrong fs type` означает, что, возможно, был неверно указан тип файловой системы для монтирования. Если вы вообще не указали тип, *mount* попытается угадать тип файловой системы по суперблоку (это справедливо только для файловых систем *minix*, *ext2*, *ext3* и *iso9660*). Если *mount* все же не может определить тип файловой системы, она опробует те типы, драйверы которых включены в ядро (перечислены в */proc/filesystems*). Если и это не приведет к успеху, команда `mount` завершит работу с ошибкой.

`device already mounted` означает, что устройство уже смонтировано в другом каталоге. Узнать, какие устройства смонтированы и где, можно с помощью команды *mount* без параметров:

```
rutabaga# mount
        /dev/hda2 on / type ext3 (rw)
        /dev/hda3 on /windows type vfat (rw)
        /dev/cdrom on /cdrom type iso9660 (ro)
        /proc on /proc type proc (rw,none)
```

Здесь видно, что имеются два раздела на жестких дисках (один с файловой системой *ext3*, а другой с файловой системой *vfat*), CD-ROM, смонтированный в каталоге */cdrom*, а также файловая система */proc*. Последнее поле каждой строки

(например, `(rw)`) перечисляет параметры, с которыми была смонтирована данная файловая система. Подробнее об этом будет рассказано далее. Обратите внимание на то, как был смонтирован CD-ROM в каталог */cdrom*. Если вы часто пользуетесь CD-ROM, то удобно создать каталог */cdrom* и монтировать в него устройство. Каталог */mnt* обычно используется для временного монтирования файловых систем, таких как гибкие диски.

Сообщение *mount-point* `busy` несколько необычно. В сущности, оно означает, что в *точке монтирования* (*mount-point*) происходит какая-то активность, не позволяющая монтировать в ней файловую систему. Обычно это связано с тем, что в данном каталоге есть открытый файл или у некоторого процесса рабочий каталог находится ниже точки монтирования. При использовании команды *mount* убедитесь, что текущий каталог в оболочке суперпользователя не находится внутри точки монтирования; выполните команду *cd /*, чтобы выйти в корневой каталог. Либо может оказаться, что в той же точке монтирования уже смонтирована другая файловая система. Выполните команду *mount* без аргументов, чтобы проверить, в чем дело.

Конечно, сообщение `other error` не очень содержательно. Есть и другие ситуации, в которых `mount` может завершиться с ошибкой. Если в рассматриваемой файловой системе есть ошибки данных или носителя, `mount` может сообщить о невозможности прочесть *суперблок* (*superblock*) файловой системы, который (в UNIX-подобных системах) является частью файловой системы, хранящей данные о файлах и атрибутах для файловой системы в целом. При попытке монтировать CD-ROM или гибкий диск и отсутствии при этом в соответствующем приводе CD-ROM или дискеты вы получите сообщение об ошибке типа:

```
mount: /dev/cdrom is not a valid block device
```

Дискеты особенно склонны к физическим дефектам (больше, чем вам может изначально показаться), а CD-ROM страдают от пыли, царапин и отпечатков пальцев, а также вставки рабочей стороной вверх и тому подобного. (При попытке монтирования Stan Rogers CD в качестве ISO 9660 вы можете столкнуться со схожими проблемами.)

Убедитесь также в существовании точки монтирования (например, */mnt*), которую вы пытаетесь использовать. Если ее нет, можно просто создать ее командой *mkdir*.

Проблемы при монтировании или доступе к файловой системе могут быть вызваны повреждением данных. В Linux есть ряд программных средств, позволяющих исправлять некоторые файловые системы; см. раздел «Проверка и исправление файловых систем» далее в этой главе.

При начальной загрузке система автоматически монтирует некоторые файловые системы. Это определяется файлом */etc/fstab*, где имеются записи для всех файловых систем, которые должны монтироваться при начальной загрузке системы. Все строки этого файла имеют формат:

```
device
mount-point
type
options
```

В данном случае *device*, *mount-point* и *type* эквивалентны по своему значению параметрам команды `mount`, а *options* представляет собой разделенный запятыми список опций ключа `-o` для команды `mount`.

Ниже приводится пример содержимого файла */etc/fstab*:[1]

```
# device      directory     type     options
/dev/hda2     /             ext3     defaults
/dev/hda3     /windows      vfat     defaults
/dev/cdrom    /cdrom        iso9660  ro
/proc         /proc         proc     none
/dev/hda1     none          swap     sw
```

Последняя строка файла описывает раздел подкачки. Описание вы найдете в разделе «Управление пространством свопинга» далее в этой главе.

На странице справочного руководства *mount*(8) перечислены возможные значения *options*. Если необходимо задать более одного параметра, их нужно перечислить через запятую без пробелов, например:

```
/dev/cdrom /cdrom iso9660 ro,user
```

Параметр `user` допускает возможность монтирования файловой системы обычными пользователями, не обладающими привилегиями суперпользователя. Если этот параметр указан, пользователи смогут монтировать устройства с помощью таких команд, как

```
mount /cdrom
```

Обратите внимание: если в команде указывается только устройство или только точка монтирования (то есть один из аргументов команды отсутствует), предварительно будет произведен поиск устройств или точек монтирования в файле */etc/fstab*, и соответствующее устройство будет смонтировано в заданном каталоге с определенными в файле параметрами. Это облегчает операцию монтирования устройств, перечисленных в */etc/fstab*.

Для большинства файловых систем следует использовать параметр `defaults`, который включает в себя ряд других параметров, таких как `rw` (доступ для чтения и записи), `async` (асинхронная буферизация в памяти ввода-вывода в файловой системе) и другие. Если только у вас нет особой нужды модифицировать один из этих параметров, используйте для большинства файловых систем значения `defaults` и `ro` для устройств, работающих в режиме только для чтения, таких как CD-ROM. Другим полезным параметром является `umask`, позволяющий задать маску по умолчанию для битов прав доступа, что бывает особенно удобно при использовании чужеродных файловых систем.

Команда *mount -a* смонтирует все файловые системы, перечисленные в */etc/fstab*. Эта команда выполняется при начальной загрузке одним из сценариев, находящихся в */etc/rc.d*, например *rc.sysinit* (или другим, в зависимости от того, где

---

[1] В общем для всех дистрибутивов в синтаксисе */etc/fstab* могут указываться еще два дополнительных необязательных параметра, уточняющих особенности монтирования. В некоторых дистрибутивах, например Mandriva 2006, по умолчанию можно видеть все шесть параметров *fstab*. Подробнее обо всех возможных параметрах можно узнать, запустив команду `man fstab`. – *Примеч. науч. ред.*

ваш дистрибутив хранит файлы с настройками). Благодаря этому все файловые системы, перечисленные в */etc/fstab*, станут доступными после запуска системы: разделы жесткого диска, привод CD-ROM и другие – все они будут смонтированы.

Исключение составляет *корневая файловая система* (*root filesystem*). Корневая файловая система, монтируемая в каталог /, обычно содержит файл */etc/fstab* и сценарии из */etc/rc.d*. Для обеспечения доступа к ним ядро должно само смонтировать корневую файловую систему во время начальной загрузки. Устройство, содержащее корневую файловую систему, должно быть прошито в образе ядра, что можно сделать командой *rdev* (см. раздел «Использование загрузочной дискеты» в главе 17). Во время начальной загрузки ядро пытается установить это устройство в качестве корневой файловой системы, последовательно перебирая ряд типов файловых систем. При загрузке ядро может вывести сообщение типа:

```
VFS: Unable to mount root fs
```

Оно может быть вызвано следующими причинами:

- В ядре прошито неверное корневое устройство.
- В ядре не скомпилирована поддержка того типа файловой системы, который имеет корневое устройство. (Подробности см. в разделе «Сборка ядра» главы 18. Обычно это происходит только при сборке собственного ядра.)
- Корневое устройство неким образом повреждено.

В любом из этих случаев ядро не может продолжить работу и впадает в панику. О том, что делать в подобных ситуациях, рассказывается в разделе «Действия в аварийных ситуациях» главы 27. Если проблема состоит в повреждении файловой системы, обычно это поправимо (раздел «Проверка и исправление файловых систем» далее в этой главе).

Чтобы быть смонтированной, файловая система не обязательно должна быть указана в */etc/fstab*, но она должна быть указана там для автоматического монтирования командой *mount -a* или в случае необходимости использовать параметр `user`.

Автоматическое монтирование устройств

Если необходимо организовать доступ к нескольким файловым системам, особенно сетевым, вас может заинтересовать достаточно новая особенность ядра Linux: автомонтировщик (*automounter*). Это комбинация функций ядра, демона и некоторых файлов с настройками, автоматически определяющих потребность в доступе к определенной файловой системе и прозрачно монтирующих эту файловую систему. Если файловая система в течение некоторого времени не используется, автомонтировщик автоматически размонтирует ее, чтобы сберечь такие ресурсы, как оперативная память и пропускная способность сети.

При намерении пользоваться автомонтировщиком необходимо, прежде всего, включить эту функцию при сборке ядра (см. раздел «Сборка ядра» в главе 18). Понадобится также включить параметр NFS.

Далее необходимо запустить демон *automount*. Это совершенно новая функция, и в вашем дистрибутиве ее может не быть. Посмотрите в каталоге */usr/lib/autofs*, и если ее там нет, нужно получить пакет *autofs* из доступного вам архива Linux, скомпилировать и установить его согласно прилагаемым инструкциям.

Имейте в виду, что существуют две версии поддержки автомонтирования: версия 3 и версия 4. В большинство дистрибутивов по-прежнему включается версия 3, поэтому ее мы и опишем.

Можно автоматически монтировать файловые системы в любом месте, но для простоты мы предполагаем, что вы хотите автоматически монтировать все файловые системы в одном каталоге, который мы назовем */automount*. Если вы хотите, чтобы точки автоматического монтирования были разбросаны по всей файловой системе, необходимо будет использовать несколько демонов `automount`.

Если вы сами скомпилировали пакет *autofs*, неплохо начать с копирования примеров файлов конфигурации, которые можно найти в каталоге *sample*, и приспособить их для своих нужд. Скопируйте файлы *sample/auto.master* и *sample/auto.misc* в каталог */etc*, а файл *sample/rc.autofs* под именем *autofs* в то место, где ваш дистрибутив хранит сценарии начальной загрузки. Допустим, что это */etc/init.d*. (К сожалению, в некоторых дистрибутивах файлы примеров отсутствуют даже при наличии пакета *autofs*. В этом случае можно порекомендовать загрузить оригинальную версию пакета.)

Сначала нужно отредактировать файл конфигурации */etc/auto.master*. В нем перечислены все каталоги (так называемые *точки монтирования*), ниже которых автомонтировщик должен монтировать разделы. Поскольку в примере этой главы мы решили использовать только один раздел, нам нужно будет сделать только одну запись, тогда содержимое файла будет выглядеть следующим образом:

```
/automount /etc/auto.misc
```

Файл состоит из строк, в каждой из которых два элемента, разделенных пробелом. Первый элемент указывает на точку монтирования, а второй элемент определяет так называемый *файл соответствия* (*map file*), в котором указано, как и когда должны быть смонтированы автоматически монтируемые устройства или разделы. Такой файл соответствия должен существовать для каждой точки монтирования.

В нашем случае файл */etc/auto.misc* содержит следующие строки:

```
cd          -fstype=iso9660,ro      :/dev/scd0
floppy      -fstype=auto            :/dev/fd0
```

Этот файл состоит из элементов, каждый из которых определяет устройство или раздел для автоматического монтирования. Строки состоят из двух обязательных и одного дополнительного полей, разделенных пробелами. Первое поле – обязательный элемент, он указывает имя каталога, в который должны монтироваться устройство или раздел. Это значение добавляется к точке монтирования, таким образом, CD-ROM автоматически будет смонтирован в каталог */automount/cd*.

Второе поле может не указываться, оно определяет набор флагов и параметров операции монтирования. Эти флаги эквивалентны используемым командой `mount`, за исключением типа файловой системы, который задается в виде *-fstype=*, а не *-t*.

Наконец, третье поле определяет раздел или устройство, которые должны быть смонтированы. В нашем случае мы указываем первый привод SCSI CD-ROM и первый гибкий диск соответственно. Двоеточие перед третьим полем ставить не обязательно. Оно отделяет часть, описывающую сетевой узел, от части, описывающей устройство и каталог, так же как и в команде *mount*. Поскольку эти два устройства находятся на локальной машине, слева от двоеточия ничего не указано.

При желании автоматически монтировать каталог *sources* с сервера NFS `sourcemaster` можно было оформить это примерно следующим образом:

```
sources -fstype=nfs,soft sourcemaster:/sources
```

Обратите внимание: файл */etc/auto.misc* не должен иметь прав на исполнение. Если вы сомневаетесь, дайте команду:

```
tigger# chmod a-x /etc/auto.misc
```

После редактирования файлов с настройками можно запустить демон автомонтирования, выполнив команду (замените путь соответствующим вашей системе):

```
tigger# /etc/init.d/autofs start
```

Поскольку эта команда весьма неразговорчива, нужно проверить, действительно ли запустилась служба автомонтирования. Сделать это можно так:

```
tigger# /etc/init.d/autofs status
```

Но из вывода этой команды нелегко определить, действительно ли запущено автомонтирование, поэтому лучше проверить, существует ли процесс `automount`:

```
tigger# ps -aux | grep automount
```

Если эта команда показывает процесс *automount*, все должно быть в порядке. Если нет, нужно снова проверить файлы с настройками. Может случиться, что отсутствует поддержка в ядре: либо вообще не включена поддержка автомонтирования, либо она скомпилирована в виде модуля, который не установлен. В последнем случае можно поправить дело, выполнив команду

```
tigger# modprobe autofs
```

Если это не поможет, нужно выполнить[1]

```
tigger# modprobe autofs4
```

Когда автомонтирование заработает так, как нужно, можно поместить вызовы `modprobe` и `autofs` в один из файлов с настройками, используемых при начальной загрузке системы, например в */etc/rc.local*, */sbin/init.d/boot.local* или другой файл, используемый в вашем дистрибутиве.

Если все правильно настроено, то достаточно обратиться к какому-либо каталогу ниже точки монтирования, и автомонтировщик смонтирует вам соответствующее устройство или раздел. Например, если вы введете:

```
tigger$ ls /automount/cd
```

автомонтировщик смонтирует CD-ROM, чтобы команда `ls` могла вывести его содержание. Единственная разница между обычным и автоматическим монтированием состоит в том, что при автомонтировании будет заметна небольшая задержка перед выводом содержимого каталога.

С целью сбережения ресурсов автомонтировщик размонтирует устройство или раздел, если к нему нет обращения в течение некоторого времени (по умолчанию 5 минут).

---

[1] О команде `modprobe` будет рассказано в разделе «Загружаемые драйверы устройств» главы 18.

У автомонтировщика есть несколько более тонких параметров. Например, вам не обязательно читать из файла таблицу соответствия, но можно обращаться к системным базам данных или даже позволить автомонтировщику запускать программу, результаты работы которой будут использованы в качестве таблицы соответствия. Подробности вы найдете на страницах справочного руководства *autofs*(5) и *automount*(8).

Создание файловых систем

Файловую систему можно создать командой *mkfs*. Создание файловой системы аналогично форматированию раздела или дискеты, позволяющему записывать на них файлы.

С каждой файловой системой связана своя команда *mkfs*, например, файловые системы MS-DOS могут быть созданы с помощью *mkfs.msdos*, третья расширенная файловая система – с помощью *mkfs.ext3* и т. д. Сама программа *mkfs*, позволяющая создавать файловые системы любого типа, является интерфейсом более высокого уровня к соответствующим версиям *mkfs*.

При установке Linux вы, возможно, вручную создавали файловые системы с помощью такой команды, как *mke2fs*. (Если нет, значит, программа установки сделала это за вас.) Несмотря на свое название эта команда может создавать файловые системы *ext2* и *ext3*. В действительности *mke2fs* эквивалентна команде *mkfs.ext2*. Эти программы одинаковы (а во многих системах первая из них является символической ссылкой на вторую), но обращение к *mkfs* по имени *mkfs.fstype* упрощает создание требуемой файловой системы. При отсутствии в системе команды *mkfs* можно напрямую пользоваться командами *mke2fs* или *mkfs.ext2*.

Если вы используете *mkfs*, то файловую систему можно создать с помощью следующей команды:

```
mkfs -t type device
```

где *type* – это тип создаваемой файловой системы, один из перечисленных в табл. 10.1, а *device* является устройством, на котором создается файловая система (например, */dev/fd0* для гибкого диска).

Чтобы создать файловую систему *ext2* на дискете (нет смысла создавать журналируемую файловую систему на гибком диске, именно по этой причине здесь создается файловая система *ext2*, а не *ext3*), нужно выполнить следующую команду:

```
mkfs -t ext2 /dev/fd0
```

Создать дискету MS-DOS можно с параметром *-t msdos*.

Теперь можно смонтировать дискету, как было показано в предыдущем разделе, записать на нее файлы и т. д. Не забудьте размонтировать дискету, перед тем как вынуть ее из дисковода.

При создании файловой системы уничтожаются все данные в соответствующем физическом устройстве (на дискете, в разделе жесткого диска и пр.). *mkfs* обычно не дает подсказки перед созданием файловой системы, поэтому тщательно контролируйте свои действия.

Создание файловой системы в разделе жесткого диска производится точно так же, как показано выше, за исключением того, что в качестве устройства нужно

использовать имя раздела, например */dev/hda2*. Не пытайтесь создать файловую систему на устройстве, таком, например, как */dev/hda*. Это имя относится ко всему диску, а не к отдельному его разделу. Создать раздел можно с помощью *fdisk*, как описано в разделе «Редактирование /etc/fstab» главы 2.

Следует проявлять особую осторожность при создании файловых систем на разделах жестких дисков. Тщательно проверяйте правильность аргументов *device* и *size*. При задании неправильного устройства *device* вы можете разрушить данные в текущих файловых системах, а если неверно задать *size*, можно затереть данные других разделов. Убедитесь в том, что *size* соответствует размеру раздела, определенному утилитой Linux `fdisk`.

При создании файловых систем на гибких дисках лучше всего сначала выполнить форматирование низкого уровня, помещающее на дискету информацию о секторах и дорожках, с помощью которой автоматически определяется емкость дискеты при использовании устройств */dev/fd0* или */dev/fd1*. Один из способов произвести низкоуровневое форматирование – воспользоваться командой MS-DOS `FORMAT`; другой способ – команда Linux *fdformat*. (Пользователи дистрибутива Debian должны использовать команду `superformat`.) Например, отформатировать дискету в первом дисководе можно командой:

```
rutabaga# fdformat /dev/fd0
        Double-sided, 80 tracks, 18 sec/track. Total capacity 1440 kB.
        Formatting ... done
        Verifying ... done
```

При использовании параметра `-n` команда `fdformat` пропускает этап верификации.

Каждая конкретная версия *mkfs* поддерживает несколько параметров, которые могут оказаться востребованными. Для большинства типов файловых систем поддерживается параметр `-c`, который требует выполнения проверки физического носителя на наличие сбойных блоков при создании файловой системы. Если обнаруживаются сбойные блоки, они помечаются и пропускаются при записи данных в файловую систему. Чтобы использовать параметры, специфические для типа файловой системы, включайте их в *mkfs* после параметра `-t` *type*, например:

```
mkfs -t type -c device blocks
```

Для того чтобы выяснить, какие параметры доступны, загляните в страницы справочного руководства по конкретной версии *mkfs*. (Например, для второй расширенной файловой системы смотрите описание команды `mke2fs`.)

У вас могут быть установлены не все имеющиеся типы *mkfs*. В этом случае *mkfs* не сможет создать файловую систему нужного вам типа, если для него не найдено *mkfs.<type>*. Для многочисленных типов файловых систем, поддерживаемых Linux, всегда где-то есть соответствующая программа *mkfs.<type>*.

Если возникают проблемы при использовании `mkfs`, они могут быть связаны с проблемами доступа Linux к физическому устройству. В случае дискеты это может просто означать, что она повреждена. В случае жесткого диска проблемы могут быть более серьезными. Например, могут быть проблемы чтения диска у драйвера дискового устройства, находящегося в ядре. Это может быть связано с аппаратным устройством или просто неверным заданием геометрии диска. Справочную информацию по различным версиям *mkfs* можно найти на страницах справочно-

го руководства, а рекомендации по устранению проблем, возникающих во время установки, – в главе 2, так как они применимы и в данной ситуации.[1]

Проверка и исправление файловых систем

Иногда необходимо проверить целостность файловых систем Linux и исправить их при наличии ошибок или утерянных данных. Такие ошибки обычно происходят при крахе системы или при аварийном отключении питания, когда ядро не может синхронизировать кэш файловой системы с содержимым диска. Такие ситуации случаются относительно редко. Однако при аварии системы во время записи большого файла этот файл может быть потерян, а связанные с ним сектора диска окажутся помечены как «используемые», хотя файла, который их использует, фактически не существует. В других случаях ошибки могут быть вызваны случайной записью данных непосредственно на устройство жесткого диска (например, */dev/hda*) или один из его разделов.

Программа *fsck* используется для проверки файловых систем и исправления возникающих проблем. Подобно *mkfs*, программа *fsck* служит интерфейсом для специфических в отношении файловых систем программ *fsck.type*, таких как *fsck.ext2* для второй расширенной файловой системы. (Как и *mkfs.ext2*, *fsck.ext2* является символической ссылкой на *e2fsck*, и каждую из них можно запустить непосредственно, если *fsck* не установлена.)

Пользоваться *fsck* очень просто. Команда имеет следующий формат:

```
fsck -t type device
```

где *type* – тип файловой системы, которую нужно восстановить, согласно табл. 10.1, а *device* является устройством (разделом жесткого диска или гибким диском), на котором находится файловая система.

Например, чтобы выполнить проверку файловой системы *ext3* в разделе */dev/hda2*, следует выполнить команду:

```
rutabaga# fsck -t ext3 /dev/hda2
        fsck 1.34 (25-Jul-2003)
        /dev/hda2 is mounted. Do you really want to continue (y/n)? y

        /dev/hda2 was not cleanly unmounted, check forced.
        Pass 1: Checking inodes, blocks, and sizes
        Pass 2: Checking directory structure
        Pass 3: Checking directory connectivity
        Pass 4: Checking reference counts.
        Pass 5: Checking group summary information.

        Free blocks count wrong for group 3 (3331, counted=3396). FIXED
        Free blocks count wrong for group 4 (1983, counted=2597). FIXED
        Free blocks count wrong (29643, counted=30341). FIXED
        Inode bitmap differences: -8280. FIXED
        Free inodes count wrong for group #4 (1405, counted=1406). FIXED
```

[1] Кроме того, процедура создания файловой системы ISO 9660 для CD-ROM сложнее, чем простое форматирование и копирование файлов. Подробности вы найдете в документе «CD-Writing HOWTO».

```
Free inodes count wrong (34522, counted=34523). FIXED

/dev/hda2: ***** FILE SYSTEM WAS MODIFIED *****
/dev/hda2: ***** REBOOT LINUX *****
/dev/hda2: 13285/47808 files, 160875/191216 blocks
```

Прежде всего обратите внимание, что система запрашивает подтверждение, перед тем как начать проверку смонтированной файловой системы. Если файловая система была смонтирована в момент запуска *fsck* и в процессе проверки были найдены и исправлены ошибки, систему придется перезагрузить. Это вызвано тем, что произведенные *fsck* изменения не затрагивают внутреннее представление системы о структуре файловой системы. Вообще говоря, проверять смонтированные файловые системы не рекомендуется.

В данном случае было обнаружено и исправлено несколько ошибок, а поскольку файловая система была смонтирована, система предупредила, что машину следует перезагрузить.

Как можно проверить файловую систему, не монтируя ее? За исключением корневой, любую файловую систему можно просто размонтировать с помощью команды *umount*, прежде чем запускать команду *fsck*. Однако корневую файловую систему нельзя размонтировать, пока система работает. Один из способов проверить корневую файловую систему, пока она размонтирована, – это использовать комбинацию загрузочной и корневой дискет из дистрибутива, например инсталляционных дискет. При этом корневая файловая система содержится на дискете, а проверяемая корневая файловая система (на жестком диске) остается размонтированной, благодаря чему появляется возможность ее проверки. Более подробно об этом рассказывается в разделе «Действия в аварийных ситуациях» главы 27.

Другой способ проверить корневую файловую систему – смонтировать ее в режиме только для чтения. Это можно сделать с помощью параметра `ro` в приглашении загрузки LILO (см. раздел «Определение дополнительных параметров во время загрузки» главы 17). Однако некоторые части системной конфигурации (например, программы, выполняемые */etc/init* во время загрузки) могут потребовать доступа к файловой системе для записи, поэтому нет возможности нормально загрузить систему, иначе эти программы не выполнятся. Чтобы выполнить начальную загрузку с монтированием корневой файловой системы в режиме только для чтения, можно загрузить систему в однопользовательском режиме (с помощью параметра загрузки `single`). Тем самым пропускается дополнительная настройка системы при загрузке. После этого можно проверить корневую файловую систему и перезагрузить систему обычным образом. Для этого, находясь в интерфейсном экране начального загрузчика GRUB, необходимо отредактировать команду загрузки, добавив в нее параметр `ro`.

Чтобы корневая файловая система была смонтирована в режиме только для чтения, можно либо использовать параметр загрузки `ro`, либо с помощью *rdev* установить флаг только для чтения в самом образе ядра.

Во многих системах Linux проверка файловых систем производится автоматически во время начальной загрузки. Делается это обычно путем запуска *fsck* из */etc/rc.d/rc.sysinit*. В этом случае система обычно сначала монтирует файловую систему только для чтения, запускает *fsck* для ее проверки, а затем выполняет команду:

```
mount -w -o remount /
```

Параметр *-o remount* вызывает перемонтирование указанной файловой системы с новыми параметрами. В данном случае параметр *-w* (эквивалентный *-o rw*) заставляет монтировать файловую систему для чтения-записи. В итоге корневая файловая система заново монтируется в режиме доступа для чтения и записи.

При выполнении *fsck* во время начальной загрузки программа проверяет все файловые системы, кроме корневой, прежде чем смонтировать их. По завершении работы *fsck* монтируются все прочие файловые системы с помощью команды *mount*. Посмотрите в файлах из каталога */etc/rc.d*, особенно в *rc.sysinit* (если он есть в вашей системе), как это делается. Если вы хотите отключить эту функцию в своей системе, закомментируйте строки в соответствующем файле */etc/rc.d*, которые запускают *fsck*.

Есть несколько параметров, которые можно передать конкретной версии *fsck*. Для большинства типов поддерживаются параметры: *-a*, автоматически подтверждающий все запросы, которые может делать *fsck.type*; *-c*, осуществляющий проверку сбойных блоков, как *mkfs*; *-v*, выводящий текстовые сообщения во время проверки. Эти параметры должны указываться после типа файловой системы, например:

```
fsck -t type -v device
```

запускает *fsck* с текстовыми сообщениями.

Более подробные сведения вы найдете на страницах справочного руководства для команд *fsck* и *e2fsck*.

Не для всех типов файловых систем, поддерживаемых Linux, есть свои версии *fsck*. Для проверки и исправления файловых систем MS-DOS следует использовать средства для MS-DOS, такие как Norton Utilities. Можно найти версии *fsck* по крайней мере для второй и третьей расширенных файловых систем, а также для файловых систем Reiser, JFS и Minix.[1]

В разделе «Действия в аварийных ситуациях» главы 27 мы предоставим дополнительные сведения о проверке файловых систем и восстановлении после аварий. *fsck* не обязательно обнаружит и исправит все ошибки в файловых системах, но с большинством стандартных проблем она справится. Если по ошибке был удален какой-то важный файл, то в настоящее время нет достаточно надежного способа восстановить его – по крайней мере, *fsck* это не под силу. Сейчас ведутся некоторые работы по созданию утилиты «undelete», которая восстанавливала бы удаленные файлы в файловой системе *ext2*, тем не менее обязательно создавайте резервные копии данных, представляющих определенную ценность, или хотя бы используйте команду *rm -i*, которая всегда запрашивает подтверждение перед удалением файла.

Шифрующие файловые системы

Начиная с версии ядра 2.2 Linux поддерживает шифрующие файловые системы. Однако из-за законов, регулирующих экспорт программного обеспечения, содержащего криптографические алгоритмы, данная возможность была реализо-

[1] В действительности многие дистрибутивы имеют в своем арсенале команду, которая называется *dosfsck/fsck.msdos*, но пользоваться ею не рекомендуется.

вана в виде исправлений к ядру, доступных на сайте *http://www.kerneli.org/* (обратите внимание на символ *i* в имени сайта, он означает «international», то есть «международный», и указывает, что сервер расположен за пределами Соединенных Штатов). Ныне этот сайт не поддерживается.

Начиная с версии ядра 2.4, активная поддержка исправлений *kerneli* была прекращена. Предпочтение было отдано методу шифрования файловых систем *loop-aes* (*http://loop-aes.sourceforge.net/*), который мог быть собран в виде модуля ядра, ограничивался поддержкой стандарта шифрования AES и развивался наиболее активно.[1]

С появлением ядра версии 2.6 наступил конец платформе шифрования *kerneli*, поскольку группа разработчиков ядра создала абсолютно новую платформу. С того момента данная платформа была включена в состав «ванильного» (vanilla) ядра (ядра, которым управляет сам Линус). В данной книге будет говориться только о ядрах версии 2.6, хотя инструментальные средства пространства пользователя не очень сильно изменили свои интерфейсы. Например, все команды *losetup* могут работать с ядрами от *kerneli*, но параметры команды *mount* могут отличаться.

Настройка ядра

Поддержка шифрующих файловых систем основана на использовании того, что называется *петлевым блочным устройством* (возможно, вы уже знаете, что петлевые блочные устройства могут использоваться для монтирования ISO-образов компакт-дисков, чтобы обеспечить доступ к их содержимому).

Таким образом, в настройках ядра необходимо включить параметр Device Drivers Loopback, а также параметр Cryptoloop support в том же разделе. Для доступа к устройствам Cryptoloop использует криптографический прикладной интерфейс (API) ядра версии 2.6, параметры которого настраиваются в разделе Cryptographic options. Как правило, в большинстве ситуаций достаточно включить сборку всех алгоритмов (шифрования, сжатия и вычисления контрольных сумм) в виде модулей, что в новейших ядрах определяется значениями по умолчанию. Параметр Testing module можно оставить выключенным.

После этого ядро необходимо собрать и установить, как и любое другое ядро. Если поддержка устройств Cryptoloop была выполнена в виде модуля, во время запуска системы его необходимо загрузить с помощью команды *modprobe cryptoloop*.

Последнее, что необходимо проверить, это наличие пакета *util-linux*, который может работать с криптографическим API ядра. Этот пакет содержит множество команд системного администрирования для работы с криптографической поддержкой в ядре. К сожалению, в момент написания этих строк необходимые исправления к последнему выпуску *util-linux* еще не были выпущены. Однако многие производители поставляют исправленные версии пакета в составе своих дистрибутивов. В документации к имеющемуся у вас пакету *util-linux* обязательно

---

[1] Аббревиатура «AES» происходит от «Advanced Encryption Standard» (улучшенный стандарт шифрования). В основе AES лежит алгоритм Rijndael. AES – это дальнейшее развитие алгоритма DES (Data Encryption Standard – стандарт шифрования данных), насчитывающего уже более чем 20-летнюю историю.

проверьте, поддерживается ли `cryptoapi`. Если команда `losetup` (описываемая в следующем разделе) терпит неудачу с сообщением о недопустимых параметрах, следовательно, поддержка криптографического API в дистрибутиве отсутствует. В этом случае нужно внести необходимые исправления в исходные тексты ядра и пересобрать его, как подробно описано в документе «Cryptoloop-HOWTO» (*http://www.tldp.org/HOWTO/Cryptoloop-HOWTO/*).

Создание шифрующей файловой системы

Шифрующие файловые системы могут создаваться как поверх целых разделов, так и внутри обычных файлов. Это сродни настройке пространства подкачки. Однако, чтобы как-то замаскировать области с зашифрованными данными, файл или раздел при инициализации должны заполняться случайными числами вместо нулей, например так:

```
dd if=/dev/urandom of=файл-или-раздел bs=1k count=размер-в-килобайтах
```

При заполнении раздела параметр `count` следует опустить, а сообщение об ошибке «device full» (устройство заполнено) можно просто игнорировать.

После того как основа для создания файловой системы будет подготовлена, можно создать петлевое устройство:

```
losetup -e алгоритм /dev/loop0 файл-или-раздел
```

Перечень доступных алгоритмов шифрования можно посмотреть в каталоге */proc/crypto*.

Команда попросит ввести ключевую фразу *всего один раз*. Она *не предложит* ввести фразу повторно для проверки. Фраза должна выбираться достаточно случайно, чтобы уменьшить вероятность взлома методом подбора по словарю. Мы рекомендуем создавать 128-разрядный случайный ключ для алгоритма шифрования следующим образом:

```
head -c16 /dev/random | mimencode
```

Замените параметр *-c16* на *-c32*, если необходимо получить 256-разрядный ключ. Такие ключевые фразы весьма сложны для запоминания – все-таки они будут представлять собой случайный набор символов. Запишите ключ на листке бумаги и храните его подальше от компьютера (например, в кошельке).

По завершении работы команды все, что будет записываться в устройство */dev/loop0*, будет попутно шифроваться выбранным алгоритмом и после этого записываться в назначенный файл или раздел.

Далее необходимо создать в устройстве файловую систему в устройстве */dev/loop0* точно так же, как и на любом другом устройстве. Например, чтобы создать файловую систему *ext3*, следует воспользоваться уже знакомой командой *mke2fs -j*. После того как файловая система будет создана, можно попробовать смонтировать ее:

```
mount -t ext3 /dev/loop0 точка-монтирования
```

Запишите текстовый файл в это устройство и попробуйте отыскать его содержимое в файле или разделе, который служит носителем шифрующей файловой системы, например с помощью команды `grep`. Поскольку содержимое файла было зашифровано, вам едва ли удастся отыскать его.

После размонтирования файловой системы командой *umount /dev/loop0* не забудьте удалить петлевое устройство командой *losetup -d /dev/loop0*.

Монтирование файловой системы

Естественно, создавать устройство и монтировать файловую систему вручную всякий раз, когда в этом возникает необходимость, достаточно утомительно. К счастью, команда *mount* может взять на себя все хлопоты по созданию петлевого устройства.

Для этого достаточно добавить параметр *-oencryption=алгоритм*, например так:

```
mount -t ext3 -oencryption=алгоритм файл-или-раздел точка-монтирования
```

параметр *encryption=алгоритм* можно также включить в файл */etc/fstab*, чтобы дать возможность пользователям монтировать и размонтировать свои собственные шифрующие файловые системы.

Проблемы безопасности

При работе с шифрующими файловыми системами необходимо помнить о следующих возможных проблемах:

- Смонтированные файловые системы доступны для чтения любому, кто обладает соответствующими привилегиями. Поэтому шифрующие файловые системы не должны оставаться смонтированными в периоды, когда они не используются.
- После создания файловой системы ключевую фразу уже нельзя будет изменить. Она будет внесена в ключ, используемый для шифрования. Если у вас достанет храбрости, можете попробовать изменить ключевую фразу, проделав следующий трюк: нужно создать командой `losetup` два петлевых устройства для одного и того же носителя шифрующей файловой системы, причем одно из устройств, например */dev/loop0*, должно быть открыто со старой ключевой фразой, а второе, например */dev/loop1*, – с новой. Убедитесь, что оба устройства создаются без ошибок (проверяйте устройства *попеременно*, но не запускайте их одновременно). Не забывайте: запрос на ввод ключевой фразы появится *всего один раз*. Размонтируйте файловую систему, если она была смонтирована. Теперь можно выполнить самый опасный шаг.

  С помощью команды *dd* скопируйте данные из первого петлевого устройства во второе:

  ```
  dd if=/dev/loop0 of=/dev/loop1 bs=4k
  ```

  Размер блока (параметр *bs=*) должен совпадать с размером страницы памяти ядра или с размером блока физического устройства, в зависимости от того, какой из них больше. Эта команда будет считывать блоки, которые по пути будут расшифровываться старой ключевой фразой, и тут же записывать их обратно, используя новую. И не забывайте молиться, чтобы не пропало питание во время этого процесса, а еще лучше приобретите блок бесперебойного питания.

  Используя трюк с двумя петлевыми устройствами, можно даже менять алгоритм шифрования данных.
- Самое слабое звено здесь – это ключевая фраза. Никакой алгоритм шифрования не поможет, если ключ построен на основе фразы, которую легко подоб-

рать. Англоязычный текст обладает уровнем случайности в 1,3 бита (показатель энтропии) на символ. Таким образом, чтобы гарантировать стопроцентный уровень безопасности, потребуется фраза длиной примерно 200 символов. С другой стороны, прием на основе устройства */dev/random* и команды *mimencode* позволяет обойтись фразой длиной в 40 символов, правда, эти символы будут подобраны абсолютно случайно.

Управление пространством свопинга

Пространство для свопинга (*swap space*) является общим термином, обозначающим дисковую память, используемую для увеличения имеющейся в системе оперативной памяти. В Linux пространство для свопинга используется для *страничной подкачки памяти* – процесса, при котором страницы памяти (в системах на базе процессоров Intel x86 страница памяти имеет размер 4096 байт, в других архитектурах это значение может быть иным) записываются на диск при недостатке оперативной памяти и считываются в нее обратно при необходимости. Процесс, с помощью которого осуществляется страничная организация памяти, довольно сложен, но для некоторых ситуаций он оптимизирован. Подсистема виртуальной памяти в Linux разрешает совместный доступ к памяти различным работающим программам. Например, если у вас одновременно запущено несколько экземпляров Emacs, то фактически в памяти находится один экземпляр программного кода Emacs. Кроме того, доступ к страницам памяти с исполняемым кодом программ (но не данных) обычно организуется в режиме только для чтения, и потому при свопинге эти страницы не записываются на диск. Такие страницы освобождаются из памяти и при повторном доступе считываются из исходного выполняемого файла.

Разумеется, пространство для свопинга не может полностью компенсировать недостаток оперативной памяти. Обращение к диску происходит значительно медленнее, чем к оперативной памяти, на несколько порядков. Поэтому свопинг используется преимущественно для того, чтобы одновременно запустить несколько программ, которые иначе не поместились бы в оперативной памяти. Если вы часто переключаетесь между этими программами, то заметите задержку, вызванную обменом страницами с диском.

В любом случае Linux поддерживает два вида пространства для свопинга: как отдельного дискового раздела для свопинга или как файла подкачки в файловых системах Linux. Можно иметь до 8 областей свопинга, при этом каждая область является дисковым файлом или разделом размером до 2 Гбайт (в системах, построенных на базе процессоров, отличных от архитектуры Intel, эти значения могут быть иными). Тот, кто силен в математике, без труда вычислит, что это дает до 16 Гбайт пространства для свопинга. (Если кто-либо использовал такой объем свопинга на практике, авторы хотели бы познакомиться с этим человеком вне зависимости от того, силен он в математике или нет.)

Учтите, при использовании для свопинга дискового раздела производительность может оказаться выше, поскольку блоки на диске располагаются непрерывно. Если же для свопинга используется файл, блоки могут быть разбросаны по файловой системе, что в некоторых случаях может сильно снизить производительность. Часто своп-файл используется при необходимости временно расширить пространство

для свопинга, когда система мечется от недостатка физической памяти и свопа. Своп-файлы – хороший способ расширять своп-пространство при необходимости.

Почти во всех Linux-системах используется пространство для свопинга того или иного вида, обычно в виде отдельного раздела. В главе 2 описывалось, как создается своп-раздел во время установки Linux. В данном разделе мы объясним, как добавлять или удалять своп-файлы и разделы. Если у вас есть пространство для свопинга, которое вас устраивает, этот раздел может не представлять для вас интереса.

Сколько у вас места для свопинга? Команда *free* сообщает сведения об использовании системной памяти:

```
rutabaga% free
             total       used       free     shared  buffers   cached
Mem:       1034304    1011876      22428          0    18104   256748
-/+ buffers/cache:     737024     297280
Swap:      1172724      16276    1156448
```

Все числа здесь представляют размер в блоках по 1024 байт. Здесь мы наблюдаем систему с 1 034 304 блоками (порядка 1 Гбайт) оперативной памяти, из которых в настоящее время используются 1 011 876 (немногим меньше). Обратите внимание: оперативной памяти в системе всегда больше, чем указано в колонке `total`, поскольку в это количество не включена память, используемая самим ядром для различных нужд.

В колонке `shared` указан объем оперативной памяти, совместно используемой процессами. Здесь, как мы видим, для этих нужд не было занято ни одного блока. В колонке `buffers` показан объем памяти, используемой в буферных кэшах ядра. Буферный кэш (о котором упоминалось в предыдущем разделе) применяется для ускорения дисковых операций, позволяя обслуживать чтение и запись на диск с помощью оперативной памяти. Буферный кэш меняет свой размер в зависимости от использования памяти в системе. Если память нужна приложениям, она отбирается у кэша. Поэтому, хотя мы видим, что выделено порядка 1 Гбайт системной памяти, не вся она (но большая часть) используется прикладными программами. В колонке `cached` указано, сколько блоков памяти помещено в кэш для последующего быстрого доступа к данным.

Поскольку память, используемую для буферов и кэша, можно передать приложениям, во второй строке (`-/+ buffers/cache`) указано количество памяти, которое реально используется приложениями (колонка `used`) или доступно для них (колонка `free`). Для получения двух чисел во второй строке суммарный объем, занятый буферами и кэшем согласно первой строке, вычитается из общей использованной памяти и складывается с объемом свободной памяти.

В третьей строке указан общий объем свопа – 1 172 724 блоков (около 1,1 Гбайт). В нашем случае использована незначительная часть свопа, поскольку оперативной памяти много. Если запустить дополнительные приложения, для них будет использовано больше буферной памяти. К пространству для свопинга система обычно прибегает в последнюю очередь, когда невозможно получить физическую память другими способами.

Обратите внимание, что *free* указывает несколько меньший объем свопа, чем сумма размеров ваших файлов и разделов для свопинга. Это связано с необходимостью в каждой области использовать несколько блоков для хранения карты,

показывающей, как будет использоваться каждая страница. Эти накладные расходы достаточно невелики и составляют лишь несколько килобайт в каждой области свопинга.

Если вы собираетесь создавать файл для свопинга, команда *df* сообщит вам сведения о свободном пространстве, которое есть в ваших файловых системах. Она выводит список файловых систем с указанием объема и текущего процента использования.

Создание пространства для свопинга

Чтобы добавить новое пространство для свопинга, нужно сначала создать файл или раздел, в котором будет размещаться своп. Если вы хотите создать раздел, это можно сделать утилитой `fdisk`, как описано в разделе «Редактирование /etc/fstab» главы 2.

Чтобы создать файл свопинга, нужно открыть файл и записать в него столько байтов, сколько должно быть в свопе, который вы хотите создать. Это можно легко сделать с помощью команды `dd`. Например, чтобы создать своп-файл размером 32 Мбайт, можно выполнить команду:

```
dd if=/dev/zero of=/swap bs=1024 count=32768
```

В результате 32 768 блоков (32 Мбайт) данных с устройства */dev/zero* будут записаны в файл */swap*. (*/dev/zero* является особым устройством, для которого операция чтения всегда возвращает нулевые байты. Оно некоторым образом противоположно устройству */dev/null*.) После создания этого файла рекомендуется выполнить команду `sync`, чтобы синхронизировать файловые системы на случай краха машины.

Создав своп-файл или раздел, можно с помощью команды `mkswap` «отформатировать» область для свопинга. Как описано в разделе «Создание пространства для свопинга» главы 2, команда `mkswap` имеет следующий формат:

```
mkswap -c device size
```

где *device* является именем своп-раздела или файла, а *size* – размер своп-области в блоках (блок по-прежнему составляет 1 Кбайт). Обычно размер не нужно указывать, поскольку `mkswap` может сама определить размер раздела. Ключ *-c* не является обязательным, он указывает на необходимость выявления сбойных блоков во время форматирования.

Например, для своп-файла, созданного в предыдущем примере, нужно выполнить команду:

```
mkswap -c /swap 32768
```

Если область для свопинга является разделом, нужно указать его имя (например, */dev/hda3*) и размер, а также в блоках.

При использовании своп-файла (но не раздела) необходимо изменить права доступа к нему, например:[1]

[1] 0600: «доступ только для *root* (то есть для системных служб) и только по чтению и записи». Всем остальным пользователям системы доступ запрещен. – *Примеч. науч. ред.*

```
chmod 0600 /swap
```

Обработав своп-файл командой `mkswap`, выполните команду `sync`, чтобы гарантировать физическую запись данных форматирования в новый своп-файл. При форматировании своп-раздела выполнять *sync* не обязательно.

Активация пространства для свопинга

Чтобы начать использовать новое пространство для свопинга, нужно активировать его командой *swapon*. Например, создав предыдущий своп-файл с помощью *mkswap* и *sync*, можно выполнить команду:

```
swapon /swap
```

Тем самым новая область свопинга добавляется к общему объему пространства для свопинга. С помощью команды *free* убедитесь, что это действительно произошло. Если вы используете для свопинга новый раздел с именем */dev/hda3*, активируйте его командой типа:

```
swapon /dev/hda3
```

Подобно файловым системам своп-области автоматически активируются при загрузке системы командой *swapon -a* в одном из стартовых файлов системы (обычно */etc/rc.d/rc.sysinit*). Эта команда просматривает файл */etc/fstab*, в котором, как вы помните из раздела «Монтирование файловых систем», содержатся данные о файловых системах и областях для свопинга. Команда *swapon -a* активирует все записи из */etc/fstab*, в поле *options* которых стоит `sw`.

Поэтому если в файле */etc/fstab* будут содержаться следующие записи:

```
# device      directory   type     options
/dev/hda3     none        swap     sw
/swap         none        swap     sw
```

то во время загрузки системы будут активированы две области для свопинга – */dev/hda3* и */swap*. Для каждой новой области свопинга нужно создать свою запись в */etc/fstab*.

Отключение пространства для свопинга

Как часто бывает, действие проще отменить, чем выполнить. Чтобы отключить пространство для свопинга, выполните команду:

```
swapoff device
```

где *device* является именем своп-раздела или файла, который вы хотите отключить. Например, отключение свопинга на устройстве */dev/hda3* производится командой:

```
swapoff /dev/hda3
```

Для отключения своп-файла можно просто удалить его командой *rm* после выполнения *swapoff*. Не удаляйте своп-файл до его отключения, это может вызвать катастрофические последствия.

Если вы отключили своп-раздел с помощью *swapoff*, то можете дальше использовать этот раздел по своему усмотрению или удалить с помощью *fdisk*.

Если своп-области соответствует запись в */etc/fstab*, удалите эту запись, иначе при очередной перезагрузке системы вы получите сообщения об ошибках из-за того, что области свопинга не будут обнаружены.

Файловая система /proc

Системы UNIX проделали долгий путь в отношении обеспечения единообразия интерфейсов различных частей системы. Как уже говорилось в главе 4, аппаратные устройства представлены в UNIX в виде файлов особого типа, расположенных в каталоге */dev*. Более детально об этом каталоге мы будем рассказывать в разделе «Файлы устройств» далее в этой главе. Однако существует особая файловая система */proc*, которая делает дальнейший шаг: она унифицирует файлы и процессы.

С точки зрения пользователя или системного администратора файловая система */proc* выглядит как любая другая файловая система; можно перемещаться по ней с помощью команды *cd*, выводить содержимое каталогов командой *ls* и просматривать содержимое файлов командой *cat*. Однако ни один из этих каталогов и файлов не занимает пространства на жестком диске. Ядро перехватывает обращения к файловой системе */proc* и «на лету» создает содержимое каталогов и файлов. Иными словами, всякий раз, когда вы выводите содержимое каталога или файла в файловой системе */proc*, ядро динамически создает содержимое, которое вы хотите увидеть.

Чтобы сделать наш рассказ менее абстрактным, рассмотрим несколько примеров. Следующий пример – это список файлов каталога верхнего уровня файловой системы */proc*:

```
tigger # ls /proc
.     3759  5538  5679  5750  6137  9           filesystems  net
..    3798  5539  5681  5751  6186  966         fs           partitions
1     3858  5540  5683  5754  6497  acpi        ide          scsi
10    3868  5541  5686  5757  6498  asound      interrupts   self
11    3892  5542  5688  5759  6511  bluetooth   iomem        slabinfo
1138  3898  5556  5689  5761  6582  buddyinfo   ioports      splash
14    4     5572  5692  5800  6720  bus         irq          stat
15    4356  5574  5693  5803  6740  cmdline     kallsyms     swaps
1584  4357  5579  5698  5826  6741  config.gz   kcore        sys
1585  4368  5580  5701  5827  6817  cpufreq     kmsg         sysrq-trigger
1586  4715  5592  5705  5829  6818  cpuinfo     loadavg      sysvipc
16    4905  5593  5706  5941  6819  crypto      locks        tty
17    5     5619  5707  6     6886  devices     mdstat       uptime
18    5103  5658  5713  6063  689   diskstats   meminfo      version
19    5193  5661  5715  6086  6892  dma         misc         vmstat
2     5219  5663  5717  6107  6894  dri         mm
2466  5222  5666  5740  6115  6912  driver      modules
2958  5228  5673  5741  6118  7     execdomains mounts
3     5537  5677  5748  6130  8     fb          mtrr
```

В вашей системе числа будут иными, но общая структура сохранится. Все эти числа являются каталогами, представляющими выполняющиеся в вашей системе процессы. Например, взглянем на информацию о процессе с ID 3759:

```
tigger # ls /proc/3759
```

```
.     auxv     delay    fd           mem      oom_score  statm   wchan
..    cmdline  environ  mapped_base  mounts   root       status
attr  cwd      exe      maps         oom_adj  stat       task
```

(Если вы используете другую версию ядра Linux, результат работы команды у вас может несколько отличаться.) Вы видите ряд файлов, каждый из которых содержит данные об этом процессе. Например, файл *cmdline* показывает командную строку, которая запустила этот процесс. *status* дает сведения о внутреннем состоянии процесса, а *cwd* содержит ссылку на текущий рабочий каталог этого процесса.

Возможно, данные об устройствах покажутся вам еще более интересными, чем данные о процессах. Все сведения, которые ядро собрало о ваших устройствах, собраны в файловой системе */proc*, хотя найти в ней требуемое может быть непросто.

Начнем с проверки памяти вашей машины. Информацию о ней можно найти в файле */proc/meminfo*:

```
owl # cat /proc/meminfo
MemTotal:      1034304 kB
MemFree:        382396 kB
Buffers:         51352 kB
Cached:         312648 kB
SwapCached:          0 kB
Active:         448816 kB
Inactive:       141100 kB
HighTotal:      131008 kB
HighFree:          252 kB
LowTotal:       903296 kB
LowFree:        382144 kB
SwapTotal:     1172724 kB
SwapFree:      1172724 kB
Dirty:             164 kB
Writeback:           0 kB
Mapped:         294868 kB
Slab:            38788 kB
Committed_AS:   339916 kB
PageTables:       2124 kB
VmallocTotal:   114680 kB
VmallocUsed:     78848 kB
VmallocChunk:    35392 kB
HugePages_Total:     0
HugePages_Free:      0
Hugepagesize:     4096 kB
```

Если вы выполните команду *free*, то увидите ту же самую информацию, только формат чисел будет несколько иной. *free*, в сущности, читает */proc/meminfo* и несколько преобразует вывод.

Большая часть системных утилит, сообщающих данные об аппаратных средствах, делают то же самое. Файловая система */proc* является переносимым и простым способом получения этих данных. Особенно полезны сведения при добавлении к системе новых устройств. Например, аппаратные платы, как правило, требуют нескольких адресов ввода-вывода для связи с процессором и операционной системой. Если вы сконфигурировали две платы так, что они используют одни

и те же адреса ввода-вывода, катастрофа неминуема. Избежать ее можно, если проверить, какие адреса ввода-вывода ядро уже определило как используемые:

```
tigger # more /proc/ioports
0000-001f : dma1
0020-0021 : pic1
0040-005f : timer
0060-006f : keyboard
0070-0077 : rtc
0080-008f : dma page reg
00a0-00a1 : pic2
00c0-00df : dma2
00f0-00ff : fpu
0170-0177 : ide1
01f0-01f7 : ide0
02f8-02ff : serial
0376-0376 : ide1
0378-037a : parport0
03c0-03df : vesafb
03f6-03f6 : ide0
03f8-03ff : serial
0cf8-0cff : PCI conf1
c000-cfff : PCI Bus #02
  c000-c0ff : 0000:02:04.0
    c000-c00f : advansys
  c400-c43f : 0000:02:09.0
    c400-c43f : e100
d000-d00f : 0000:00:07.1
  d000-d007 : ide0
  d008-d00f : ide1
d400-d4ff : 0000:00:07.5
  d400-d4ff : AMD AMD768 - AC'97
d800-d83f : 0000:00:07.5
  d800-d83f : AMD AMD768 - Controller
dc00-dcff : 0000:00:09.0
e000-e003 : 0000:00:00.0
```

Теперь вы можете определить свободные адреса ввода-вывода. Конечно, ядро может показать адреса ввода-вывода только для плат, которые оно обнаружило и распознало, но в нормально сконфигурированной системе это должно быть правилом для всех плат.

Можно также использовать файловую систему */proc* для получения другой информации, необходимой для конфигурирования новых устройств: в файле */proc/interrupts* перечислены занятые линии прерываний (IRQ), а в */proc/dma* – используемые каналы прямого доступа к памяти (DMA).

Файлы устройств

Файлы устройств позволяют программам пользователей обращаться к аппаратным устройствам через ядро системы. Они не являются файлами как таковыми, но с точки зрения программы выглядят как файлы: можно читать их, осуществлять в них запись, отображать их в память при помощи системного вызова

mmap( ) и т. д. При обращении к такому «файлу» устройства ядро распознает запрос ввода-вывода и передает его драйверу устройства, который выполняет некоторую операцию, например чтение данных из последовательного порта или отправку данных звуковой карте.

Файлы устройств (несмотря на неудачное название, мы будем продолжать использовать этот термин) предоставляют удобный способ доступа к ресурсам системы без необходимости для прикладного программиста знать, как работает само устройство. В Linux, как и в большинстве UNIX-систем, драйверы устройств сами являются частью ядра. В разделе «Сборка ядра» главы 18 мы покажем, как собрать собственное ядро, включив в него драйверы только тех устройств, которые есть в вашей системе.

Почти во всех UNIX-подобных системах файлы устройств располагаются в каталоге */dev*. Для каждого устройства, имеющегося в системе, должен быть соответствующий элемент в */dev*. Например, */dev/ttyS0* соответствует первому последовательному порту, известному в MS-DOS как COM1; */dev/hda2* соответствует второму разделу первого диска IDE. Фактически в каталоге */dev* будут содержаться записи и для отсутствующих устройств. Файлы устройств создаются обычно во время установки системы и включают в себя все возможные драйверы устройств. Они не обязательно соответствуют фактически имеющимся в компьютере аппаратным устройствам.

В */dev* есть ряд псевдоустройств, не соответствующих какой-либо реальной периферии. Например, */dev/null* действует как сливная труба для байтов. Все попытки записи в */dev/null* будут выполняться успешно, но записываемые данные при этом будут игнорироваться. Аналогичное ему устройство */dev/zero* уже демонстрировалось выше, когда обсуждался вопрос создания своп-файла: всякое чтение из */dev/zero* просто возвращает нулевые байты.

Если с помощью команды *ls -l* вывести список файлов устройств в */dev*, то получится примерно следующее (в случае использования версии *ls*, которая выделяет цветом разные типы файлов, вы должны увидеть, что файл */dev/hda* будет окрашен другим цветом, поскольку это необычный файл):

```
brw-rw----  1 root disk 3, 0 2004-04-06 15:27 /dev/hda
```

Это устройство */dev/hda*, соответствующее первому диску IDE.[1] Прежде всего, обратите внимание на первую букву `b` в поле прав доступа, означающую, что это файл блочного устройства. (У обычных файлов в первой колонке выводится символ `-`, у каталогов – `d`, и т. д.; подробнее об этом мы поговорим в следующей главе). Файлы устройств обозначаются либо символом `b` для блочных устройств, либо символом `c` для символьных устройств. Блочное устройство обычно является периферийным устройством, таким как жесткий диск, обмен данными с которым производится целыми блоками (при этом размер блока определяется устройством и не обязательно равен 1024 байт – типичному размеру блока в Linux),

[1] Диск SCSI (не столь распространенный, как IDE) будет отображаться, например, как */dev/sda*, но в новых моделях компьютеров, в которых диск стандартно подключается через интерфейс SATA (serial ATA), драйвер моделирует его как SCSI-устройство, и такой IDE-диск будет отображаться как */dev/sda*. – *Примеч. науч. ред.*

а доступ к устройству может осуществляться произвольно. Напротив, чтение и запись на символьные устройства обычно производятся последовательно, а ввод-вывод может осуществляться по одному байту. Примером символьного устройства является последовательный порт.

Обратите внимание также на то, что поле размера файла в выводе *ls -l* заменено двумя числами, разделенными запятой. Первое число – *основной номер устройства* (*major device number*), второе число – *дополнительный номер устройства* (*minor device number*). При обращении к файлу устройства из программы ядро получает запрос ввода-вывода в терминах основного и дополнительного номеров устройства. Основной номер обычно указывает на конкретный драйвер ядра, а дополнительный – на конкретное устройство, с которым работает этот драйвер. Например, у всех последовательных портов один и тот же основной номер, но разные дополнительные номера. Ядро использует основной номер для перенаправления запроса ввода-вывода соответствующему драйверу, а драйвер использует дополнительный номер для определения того, к какому конкретно устройству нужно обратиться. В некоторых случаях дополнительные номера используются для доступа к особым функциям устройства.

Правила именования файлов в */dev* представляют собой, грубо говоря, полную неразбериху.[1] Поскольку самому ядру безразлично, какие имена файлов используются в */dev* (ему нужны только основные и дополнительные номера), создатели дистрибутивов, прикладные программисты и разработчики драйверов устройств могут произвольно выбирать имена файлов устройств.

Часто один разработчик драйвера предлагает для устройства некоторое имя, которое затем изменяется, чтобы позволить разместить другие аналогичные устройства. Это может вызывать путаницу и непоследовательность по мере развития системы. Будем надеяться, что вы не столкнетесь с этой проблемой, если только не станете работать с более новыми драйверами устройств, которые находятся в стадии тестирования. Проект, получивший название `udev`, вскоре должен решить проблему конфликтов имен файлов устройств.

В любом случае файлы устройств, включенные в ваш дистрибутив, должны быть корректными для включенных в этот дистрибутив версий ядра и драйверов устройств. При обновлении ядра или добавлении драйверов устройств (см. раздел «Сборка нового ядра» главы 18) вам может понадобиться добавить файл устройства с помощью команды `mknod`. Эта команда имеет следующий формат:

```
mknod -m permissions name type major minor
```

где:

- *`name`* является полным именем пути для создаваемого устройства, например */dev/rft0*.
- *`type`* может принимать значение `c` для символьного устройства или `b` для блочного устройства.

---

[1] В некоторых моделях компьютеров, оснащенных аппаратным контроллером UDMA IDE (например, PromiceTechnology), типовые IDE-диски могут отобразиться как */dev/hdc*, */dev/hdd* и т. д., при том что */dev/hda* и */dev/hdb* будут отсутствовать. – *Примеч. науч. ред.*

- *major* является основным номером устройства.
- *minor* является дополнительным номером устройства.
- -m *permissions* является необязательным аргументом, устанавливающим биты прав доступа для нового устройства.

Допустим, вы добавляете к ядру новый драйвер устройства, и в документации сказано, что нужно создать блочное устройство */dev/bogus* с основным номером 42 и дополнительным номером 0. В этом случае вам следует выполнить команду:

```
mknod /dev/bogus b 42 0
```

Создание устройств еще более упрощается при использовании сценария командной оболочки */dev/MAKEDEV*, входящего во многие дистрибутивы: вам придется только задать тип нужного устройства, а *MAKEDEV* сама определит основной и дополнительный номера.

Если вы не указали аргумент -m *permissions*, то новое устройство получает права доступа, как для вновь созданного файла, модифицированные вашей текущей маской, – обычно 0644. Для того чтобы изменить права доступа к файлу устройства */dev/bogus* на 0660, выполните команду

```
mknod -m 660 /dev/bogus b 42 0
```

После того как файл устройства создан, изменить права доступа к нему можно командой `chmod`.

В чем важность прав доступа к устройствам? Как и для любого файла, права доступа к файлу устройства определяют, кто может обращаться к raw-устройству и как. Как видно из предыдущего примера, файл устройства */dev/hda* имеет права доступа 0660. Это означает, что только владелец и группа файла (в данном случае – группа *disk*) могут непосредственно осуществлять чтение и запись на это устройство. (С правами доступа мы познакомим вас в разделе «Владение файлами и права доступа» главы 11.)

Как правило, нежелательно предоставлять любому пользователю право непосредственного чтения/записи на некоторые устройства, особенно относящиеся к дискам и разделам на них. В противном случае любой мог бы выполнить, скажем, команду *mkfs* на разделе диска и полностью уничтожить все данные системы.

Что касается дисков и разделов, то для того чтобы уничтожить таким способом данные, требуется право записи, но право чтения также является нарушением системы безопасности. Имея право на чтение raw-файла устройства, соответствующего дисковому разделу, пользователь может просматривать чужие файлы. Аналогично файл устройства */dev/mem* соответствует оперативной памяти системы (он обычно используется лишь в исключительных случаях при отладке). Получив право чтения, достаточно грамотный пользователь может проследить за паролями регистрации других пользователей, в том числе суперпользователя, когда они вводятся при входе в систему.

Следите за тем, чтобы права доступа к любому устройству, добавляемому вами в систему, соответствовали тому, как могут и должны обращаться к нему пользователи. К таким устройствам, как последовательные порты, звуковые карты и виртуальные консоли, простые пользователи обычно могут обращаться без опасения причинить вред, однако к большинству других устройств доступ должен

быть предоставлен только суперпользователю (и программам, выполняемым с его правами).

Основное правило разграничения прав доступа к файлам устройств, которому следуют создатели большинства дистрибутивов, заключается в том, чтобы определять суперпользователя в качестве владельца файла, но использовать при этом не группу *root*, а какую-нибудь другую. Например, в дистрибутиве SUSE файл устройства */dev/video0*, который является точкой доступа к первому видеоустройству (например, телевизионной карте), принадлежит пользователю *root*, а в качестве группы выбрана группа *video*. Таким образом, можно добавить всех пользователей, кто должен иметь доступ к этому устройству, в группу *video* и тем самым гарантировать им такую возможность. Всем остальным пользователям (кроме *root*, конечно) доступ к данной телевизионной карте будет закрыт, и они не смогут просматривать телепередачи.[1]

Многие файлы в */dev* являются в действительности символическими ссылками на другие файлы устройств, созданными обычным образом с помощью команды *ln -s*. Ссылки упрощают доступ к некоторым устройствам благодаря использованию более привычных имен. Например, если у вас есть мышь для последовательного порта, доступ к ней можно получить через один из файлов устройств */dev/ttyS0*, */dev/ttyS1*, */dev/ttyS2* или */dev/ttyS3*, в зависимости от того, к какому порту подключена мышь. Многие пользователи часто создают ссылку с именем */dev/mouse* на соответствующее устройство последовательного порта, например:

```
ln -s /dev/ttyS2 /dev/mouse
```

Благодаря этому можно обращаться к */dev/mouse*, а не пытаться вспомнить, к какому порту она подключена. Это соглашение используется и для таких устройств, как *dev/cdrom* и */dev/modem*. Обычно эти файлы являются символическими ссылками на файлы устройств в */dev*, соответствующие фактическим устройствам CD-ROM или модема.

Чтобы удалить файл устройства, можно просто выполнить команду `rm`, например:

```
rm /dev/bogus
```

Удаление файла устройства не означает, что соответствующий устройству драйвер удаляется из памяти или ядра. Просто вы теряете возможность общения с драйвером конкретного устройства. Аналогично при добавлении файла устройства в системе не появляется новый драйвер устройства. Можно добавлять файлы устройств даже для несуществующих драйверов. Файлы устройств предоставляют просто «привязку» к драйверу некоторого устройства, на случай если этот драйвер окажется в ядре.

Запуск задач по расписанию с помощью cron

Первоначальной задачей компьютера была автоматизация рутинных задач. Если вам нужно производить резервное копирование диска регулярно в час ночи, зачем каждый раз вручную вводить команды, особенно если для этого нужно вы-

[1] Придет время, когда родители будут говорить своим детям: «Если ты не будешь помогать по дому, я исключу тебя из группы *video*». Разумеется, умные дети расколют пароль суперпользователя и ни о чем не будут беспокоиться.

лезать из постели? Нужно просто сообщить компьютеру о действиях, которые необходимо произвести, и потом забыть об этом. В UNIX-системах для автоматизации таких функций существует демон `cron`. Если сказать вкратце, то `cron` используется путем запуска команды `crontab` и ввода строк в специальном формате. В каждой строке указывается, какую команду нужно выполнить и когда.

Невидимо для вас `crontab` записывает ваши команды в файл с вашим именем пользователя, расположенный в каталоге */var/spool/cron/crontabs*. (Например, для пользователя *mdw* файл *crontab* будет называться */var/spool/cron/crontabs/mdw*.) Демон с именем *crond* регулярно читает этот файл и в нужное время выполняет команды. Один из *rc*-файлов системы запускает `crond` при начальной загрузке системы. Фактически команды с именем *cron* нет, есть только утилита *crontab* и демон `crond`.

В некоторых системах запускать *cron* разрешается только суперпользователю. В любом случае взглянем на полезную команду, которую вам может понадобиться выполнить от имени *root*, и рассмотрим, как должна выглядеть запись для выполнения задания планировщиком. Допустим, вы хотите ежедневно удалять устаревшие файлы из каталога */tmp*, служащего для хранения временных файлов, создаваемых разнообразными утилитами.

Учтите, что *cron* никогда и ничего не выводит на консоль. Все выходные сообщения и сообщения об ошибках отправляются электронной почтой пользователю, являющемуся владельцем соответствующего файла *crontab*. Можно переопределить такое поведение, задав строку `MAILTO=`*address* в файле *crontab* непосредственно перед списком заданий.

В большинстве систем содержимое */tmp* очищается при перезапуске системы, но, если система перезагружается редко, может оказаться полезным использование *cron* для проверки устаревших файлов (скажем, таких, к которым не было доступа в течение последних трех дней). Для этого потребуется следующая команда:

```
ls -l filename
```

Но как узнать, какое имя файла *filename* нужно задать? Нужно поместить эту команду внутрь команды *find*, которая просмотрит все файлы, содержащиеся в каталоге, и выполнит над каждым из них указанную вами операцию.

Сейчас мы зададим */tmp* в качестве каталога, в котором необходимо осуществлять поиск, и используем параметр *-atime* для поиска тех файлов, доступ к которым осуществлялся не позднее чем три дня назад. Параметр *-exec* означает «выполнить с каждым найденным файлом следующую команду»:

```
find /tmp \! -type d -atime +3 -exec ls -l {} \;
```

Мы просим `find` выполнить команду *ls -l*, которая просто выведет сведения о файле. (Многие часто используют аналогичную запись *crontab* для удаления файлов, но при этом легко нарушить систему безопасности.) Странная строка *{}* означает лишь способ сообщить системе: «Сделать это с каждым найденным в соответствии с критерием отбора файлом». Символы `\;` обозначают конец параметра `-exec`.

Теперь у нас есть команда, отыскивающая в */tmp* устаревшие файлы. Но нам еще нужно указать, как часто ее следует выполнять. В формате, используемом *crontab*, имеется шесть полей:

```
minute   hour   day   month   dayofweek   command
```

Заполняются поля следующим образом:

1. Минуты (от 0 до 59).
2. Час (от 0 до 23).
3. День месяца (от 1 до 31).
4. Месяц (от 1 до 12 или по названия: `jan`, `feb` и т. д.).
5. День недели (от 0 до 6, где 0 соответствует воскресенью, или по названиям: `mon`, `tue` и т. д.).
6. Команда (может состоять из нескольких слов).

На рис. 10.1 показана запись `cron` со всеми заполненными полями. Команда является сценарием, исполняемым в *sh* – оболочке Борна. Эта запись не очень реалистична: сценарий выполняется только при соблюдении всех условий в первых пяти полях. Это означает, что он будет выполняться в воскресенье, выпадающее на 15 января или 15 июля – нечасто такое случается! Поэтому наш пример не имеет особой практической ценности.

Если необходимо, чтобы команда выполнялась ежедневно в 1 час ночи, задайте 0 в поле *minute* и 1 в поле *hour*. В остальных трех полях должны стоять звездочки, что означает «каждый день каждого месяца в указанное время». Полная строка для `crontab` выглядит так:

```
0 1 * * * find /tmp -atime 3 -exec ls -l {} \;
```

Поскольку с полями, задающими время, можно делать много интересных вещей, еще немного поразвлекаемся с этой командой. Предположим, что необходимо выполнять команду только по первым числам каждого месяца. Тогда нужно сохранить первые два поля как есть, а в третье поле поместить 1:

```
0 1 1 * * find /tmp -atime 3 -exec ls -l {} \;
```

Для выполнения команды еженедельно по понедельникам верните звездочку в третье поле, а в пятом поле задайте 1 или `mon`:

```
0 1 * * mon find /tmp -atime 3 -exec ls -l {} \;
```

Можно достичь еще большей сложности, задавая в каждом поле несколько временных меток. Например, в следующей строке запятая означает «выполнять 1-го и 15-го числа каждого месяца»:

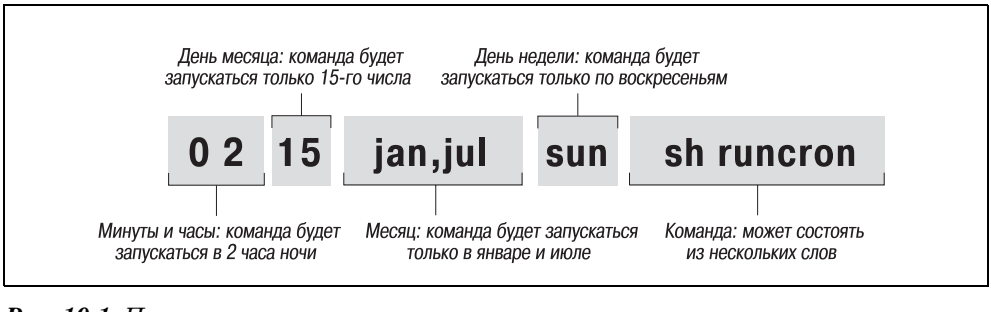


Рис. 10.1. *Пример записи cron*

```
0 1 1,15 * * find /tmp -atime 3 -exec ls -l {} \;
```

дефис означает «выполнять ежедневно с 1-го по 15-е число включительно»:

```
0 1 1-15 * * find /tmp -atime 3 -exec ls -l {} \;
```

а символ слэша, за которым следует 5, означает «выполнять каждый пятый день», то есть 1-го, 6-го, 11-го и т. д.:

```
0 1 */5 * * find /tmp -atime 3 -exec ls -l {} \;
```

Теперь можно действительно поместить запись в файл *crontab*. Зарегистрируйтесь как *root* (такого рода вещи должен делать суперпользователь) и введите команду *crontab* с ключом *-e* (от слова «edit» – редактировать):

```
rutabaga# crontab -e
```

По умолчанию эта команда начинает сеанс редактирования *vi*. Если вы предпочитаете XEmacs, это можно указать до запуска *crontab*. Для Борн-совместимой оболочки введите команду:

```
rutabaga# export VISUAL=xemacs
```

Для оболочки C:

```
rutabaga# setenv VISUAL xemacs
```

В некоторых версиях *crontab* можно вместо `VISUAL` использовать переменную окружения `EDITOR`. Введите одну-две строки, начинающиеся с решетки (`#`), в которых будут располагаться комментарии, объясняющие ваши действия, а затем введите свою запись в *crontab*:

```
# Вывести список файлов из /tmp, которые старше 3 дней.
# Запускается в 1:00 каждую ночь.
0 1 * * * find /tmp -atime 3 -exec ls -l {} \;
```

При выходе из *vi* команды будут сохранены. Посмотреть свою запись *crontab* можно, введя команду

```
rutabaga# crontab -l
```

Мы еще не говорили о важной стороне нашей записи *crontab*: а куда будет направляться вывод? По умолчанию *cron* отправляет вывод не на стандартное устройство вывода и стандартное устройство вывода ошибок, а пользователю в виде сообщения электронной почты. В данном случае почта будет отправлена пользователю *root*, но ее следует автоматически перенаправить вам как системному администратору. В */usr/lib/aliases* (или */etc/aliases* в дистрибутивах SUSE, Debian и Red Hat) должна быть следующая строка:

```
root: имя-вашей-учетной-записи
```

Чуть ниже мы покажем, как сохранить вывод в файле, а не отправлять его почтой.

Приведем пример еще одной команды, часто используемой в файлах *crontab*. Она создает резервную копию каталога на магнитной ленте. Предполагается, что перед выполнением команды в привод будет установлена лента. Сначала команда *mt* производит перемотку ленты в устройстве */dev/qft0* в начало. Затем команда `tar` переносит на ленту все файлы из каталога */src*. Точка с запятой служит разделителем команд – это стандартный синтаксис оболочки:

```
# сделать резервную копию каталога /src раз в два месяца.
0 2 1 */2 * mt -f /dev/qft0 rewind; tar cf /dev/qft0 /src
```

Первые два поля обеспечивают выполнение команды в 2:00, а третье поле указывает на первое число месяца. Четвертое поле указывает на выполнение раз в два месяца. Того же результата можно добиться более понятной командой, например:

```
0 2 1 jan,mar,may,jul,sep,nov * mt -f /dev/qft0 rewind; \
tar cf /dev/qft0 /src
```

В разделе «Создание резервных копий» главы 27 объясняется, как проводить создание резервных копий на регулярной основе.

В следующем примере каждые два дня запускается *mailq*, чтобы проверить, не застряло ли какое-нибудь письмо в очереди отправки почты, и послать письмо администратору с результатами проверки. Если почта в очереди есть, то в отчет включаются сведения об адресации и проблемах с доставкой, в противном случае сообщение пусто:

```
0 6 */2 * * mailq -v | \
mail -s ¨Проверка очереди почтовых сообщений¨ postmaster
```

Вероятно, не хотелось бы получать каждый день письмо, если все в порядке. В приведенных нами примерах команды не порождают никакого вывода, если только не происходит ошибка. Однако вы можете перенаправить стандартный вывод в */dev/null* или посылать вывод в журнал (обратите внимание на использование двух знаков >, которые позволяют избежать затирания предшествующего вывода):

```
0 1 * * * find /tmp -atime 3 -exec ls -l {} \; >> /home/mdw/log
```

В этой записи мы перенаправляем стандартный вывод, но позволяем отправлять стандартный вывод ошибок как почтовые сообщения. Это может быть очень удобно, поскольку при возникновении проблем мы получим сообщение об ошибке. Если вы не хотите получать почтовые сообщения ни при каких обстоятельствах, перенаправьте в файл как стандартное устройство вывода, так и стандартное устройство вывода ошибок:

```
0 1 * * * find /tmp -atime 3 -exec ls -l {} \; >> /home/mdw/log 2>&1
```

При сохранении вывода в файле журнала вы сталкиваетесь с проблемой неуклонного роста размера файла. Можно создать еще одну запись *cron*, которая обеспечит удаление файла, скажем, раз в неделю.

В записях *crontab* можно использовать команды только оболочки Борна. Это означает невозможность использования удобных расширений, применяемых в *bash* и других современных оболочках, например псевдонимов или знака ~ для обозначения исходного каталога пользователя. Однако можно использовать `$HOME`; `cron` распознает переменные окружения `$USER`, `$HOME` и `$SHELL`. Каждая команда выполняется из вашего исходного каталога, используемого в качестве рабочего.

Некоторые пользователи предпочитают указывать в записях *crontab* абсолютные пути к командам, например */usr/bin/find* и */bin/rm*. Тем самым всегда выбирается правильная команда независимо от корректности задания пути поиска.

Если команда становится слишком длинной и сложной, чтобы уместиться в одной строке, создайте сценарий оболочки и запускайте его через *cron*. Не забудьте

сделать сценарий исполняемым (выполните команду *chmod +x*) или запускайте его вызовом командного интерпретатора:

```
0 1 * * * sh runcron
```

Системному администратору часто приходится создавать файлы crontab для фиктивных пользователей, таких как *news* или *UUCP*. Запуск всех утилит от имени *root* был бы излишним и потенциально опасным, поэтому создаются такие специальные пользователи.

Выбор пользователя влияет также на право владения: файл crontab для *news* должен запускать файлы, владельцем которых является *news*, и т. д. В целом нужно следить, чтобы владельцем утилит был тот пользователь, от имени которого создается файл crontab.

Суперпользователь может редактировать файлы crontab других пользователей с помощью параметра -u:

```
tigger # crontab -u news -e
```

Это удобно, поскольку нельзя зарегистрироваться в системе как *news*, но может потребоваться отредактировать файл crontab для *news*.

Однократный запуск задач

Как рассказывалось в предыдущем разделе, с помощью cron можно запланировать исполнение по расписанию многократно повторяющихся заданий. Но как быть, если некоторая команда должна быть запущена один или, по крайней мере, ограниченное число раз, причем в неудобное для человека время, когда он не сможет находиться рядом с компьютером и ввести эту команду интерактивно? Само собой разумеется, такую команду можно внести в список *crontab* и удалить ее позднее или указать такие условия запуска, которые встречаются крайне редко. Но как раз для таких случаев существует специализированный инструмент, который называется at.

Команды, которые должны быть исполнены, at принимает с устройства стандартного ввода или из файла. Время запуска можно указать несколькими способами, включая способы, близкие к человеческому языку, как, например, noon (полдень), midnight (полночь) и даже, что очень интересно, teatime (чаепитие, которое для многих британских пользователей наступает в 16:00).

Для запуска команд в нужное время недостаточно иметь в системе команду at, нужно, чтобы еще был запущен демон atd. Порядок запуска демона зависит от разновидности дистрибутива, но наиболее типичны варианты: rcatd start и /etc/init.d/atd start. Но кроме этого можно запустить демон и напрямую командой */usr/sbin/atd*, обладая при этом привилегиями суперпользователя.

Для примера предположим, что необходимо загрузить из Интернета большой файл, причем сделать это нужно поздно ночью, когда стоимость доступа к Интернету самая низкая или когда линии связи менее всего загружены и потому скорость загрузки файла должна быть выше. Предположим далее, что для установления соединения с Интернетом (через коммутируемое соединение) должна быть запущена команда connectinet, а для закрытия этого соединения – команда

disconnectinet. Собственно загрузка файла должна выполняться с помощью команды wget:

```
tigger$ at midnight
warning: commands will be executed using /bin/sh
at> connectinet
at> wget ftp://overloadedserver.lotsastuff.com/pub/largefiles/reallylargefile.bz2
at> disconnectinet
at> <EOT>
job 1 at 2005-02-26 00:00
```

После ввода команды at midnight на экране сначала появится предупреждение, что все запланированные команды будут исполняться с помощью другого командного интерпретатора (для интерактивного взаимодействия с системой мы пользуемся командной оболочкой Z, тогда как команда *at* использует оболочку Борна), а затем предложит ввести команды одну за другой. По окончании ввода нужно нажать комбинацию Ctrl+D (в примере это место помечено как <EOT>). После этого at выведет *номер задания* (*job number*), а также дату и время, когда задание будет выполнено. Для вас не должно быть секретом, что команда будет исполнена, только если компьютер останется включенным!

При желании можно получить список запланированных заданий, для этого нужно воспользоваться командой *atq*:

```
tigger$ atq
1       2005-02-26 00:00 a kalle
```

Номер задания отображается в первой колонке, затем следуют дата, когда задание будет выполнено, символ, определяющий название используемой очереди (в данном случае a – иногда здесь можно увидеть нечто большее, чем простое имя очереди, но такое встречается крайне редко, поэтому мы не будем заострять ваше внимание на этом), и, наконец, владелец очереди.

Если вы вдруг поймете, что запланированное вами задание следует отменить, его в любой момент можно будет удалить, если при этом известен номер задания, который, впрочем, можно узнать с помощью команды atq, если вдруг вы забудете, какой номер сообщила команда at.

Удаление заданий из очереди выполняется с помощью команды *atrm*. Ей нужно просто передать номер задания:

```
tigger$ atrm 1
```

Команда atrm не отличается многословием, но результат ее работы всегда можно проверить с помощью команды *atq*:

```
tigger$ atq
```

Если задание с нужным номером не появилось в списке, значит, оно было удалено.

Управление системными журналами

Демон *syslogd* регистрирует в журналах различные виды активности системы, такие как отладочные сообщения sendmail и предупреждения, выводимые ядром. *syslogd* выполняется как демон и обычно запускается из какого-либо *rc*-файла во время загрузки.

Управление тем, куда *syslogd* записывает информацию, осуществляется с помощью файла */etc/syslog.conf*. Ниже приводится пример такого файла (обычно он имеет значительно более сложный вид):

```
*.info;*.notice /var/log/messages
mail.debug /var/log/maillog
*.warn /var/log/syslog
kern.emerg /dev/console
```

В первом поле каждой строки перечислены типы сообщений, которые нужно регистрировать, а во втором поле указано место, где они должны быть зарегистрированы. Первое поле имеет формат:

```
facility.level [; facility.level ... ]
```

где *facility* является системным приложением или средством, генерирующим сообщение, а *level* является уровнем важности сообщения.

Например, *facility* (средство) может принимать значение `mail` (для почтового демона), `kern` (для ядра), `user` (для программ пользователей) или `auth` (для программ аутентификации, таких как `login` или `su`). Звездочка в этом поле обозначает все средства.

level (уровень) может принимать значения (по возрастанию уровня важности): `debug`, `info`, `notice`, `warning`, `err`, `crit`, `alert` или `emerg`.

В приведенном примере файла */etc/syslog.conf* мы видели, что все сообщения уровня важности *info* и *notice* (информация и извещения) регистрируются в */var/log/messages*, все сообщения уровня *debug* (отладка) от почтового демона регистрируются в */var/log/maillog*, а все сообщения *warn* (предупредительные) регистрируются в */var/log/syslog*. Кроме того, все предупреждения ядра *emerg* (чрезвычайные) посылаются на консоль (которой является рабочая виртуальная консоль или эмулятор терминала, запущенный с параметром `-C`).

Регистрируемые демоном *syslogd* сообщения обычно содержат дату, указание процесса или средства, доставившего сообщение, и само сообщение, причем все это в одной строке. Например, сообщение ядра об ошибке данных в файловой системе *ext2fs* может появиться в журнале в виде:

```
Dec 1 21:03:35 loomer kernel: EXT2-fs error (device 3/2):
ext2_check_blocks_bit map: Wrong free blocks count in super block,
stored = 27202, counted = 27853
```

Аналогично успешное выполнение команды `su` для регистрации с именем *root* может выглядеть в журнале так:

```
Dec 11 15:31:51 loomer su: mdw on /dev/ttyp3
```

Файлы журналов могут иметь большое значение при отслеживании возникших в системе проблем. Если журнал слишком разрастается в размере, его можно очистить командой *cat /dev/null>logfile*; сам файл для ведения журнала при этом сохраняется.

Возможно, в вашей системе уже организован запуск *syslogd*, и файл */etc/syslog.conf* содержит корректные параметры настройки. Однако важно знать, где находятся ваши файлы системных журналов и какие программы в них пред-

ставлены. Если приходится регистрировать много сообщений (например, отладочные сообщения ядра, которые могут быть очень многословными), то можно отредактировать *syslog.conf* и потребовать от *syslogd* заново прочесть файл с настройками, выполнив команду:

```
kill -HUP `cat /var/run/syslog.pid`
```

Обратите внимание на использование обратных кавычек при получении идентификатора процесса *syslogd*, содержащегося в */var/run/syslog.pid*.

В системе могут существовать и другие системные журналы. В их число входят:

/var/log/wtmp

Этот файл содержит двоичные данные о времени регистрации и продолжительности работы всех пользователей системы. Он используется командой `last` для вывода списка регистрировавшихся пользователей. Вывод `last` может выглядеть так:

```
mdw      tty3     Sun Dec 11 15:25   still logged in
mdw      tty3     Sun Dec 11 15:24 - 15:25 (00:00)
mdw      tty1     Sun Dec 11 11:46   still logged in
reboot   ~        Sun Dec 11 06:46
```

В */var/log/wtmp* также добавляются записи о перезагрузке системы.

/var/run/utmp

Это еще один двоичный файл, содержащий сведения о пользователях, зарегистрированных в системе в настоящее время. Он используется такими командами, как *who*, *w* и *finger*, для вывода данных о подключенных пользователях. Например, команда *w* может вывести такие данные:

```
3:58pm up 4:12, 5 users, load average: 0.01, 0.02, 0.00
User   tty      login@   idle JCPU PCPU   what
mdw    ttyp3    11:46am    14             -
mdw    ttyp2    11:46am          1        w
mdw    ttyp4    11:46am                   kermit
mdw    ttyp0    11:46am    14             bash
```

Мы видим время регистрации каждого пользователя (в данном случае один пользователь зарегистрировался несколько раз), а также выполняемую им в данное время команду. На странице справочного руководства *w*(1) дается объяснение всех выводимых полей.

/var/log/lastlog

Этот файл аналогичен *wtmp*, но используется другими программами (такими, как *finger*) для определения времени последней регистрации пользователя.

Учтите, что формат файлов *wtmp* и *utmp* в разных системах может быть различен. Программы могут быть откомпилированы в расчете на тот или иной формат. Поэтому команды, которые используют эти файлы, могут выводить неточную информацию, особенно если файлы повреждены программой, которая пишет в них информацию в неверном формате.

Журнальные файлы могут быстро разрастаться в объеме, и при недостаточном дисковом пространстве нужно принимать меры, чтобы разделы не слишком бы-

стро заполнялись данными. Конечно, можно периодически удалять файлы журналов, но вам, возможно, не следует этого делать, поскольку в файлах журналов могут содержаться данные, представляющие ценность в критических ситуациях.

Один из способов решения проблемы – периодическое копирование журналов в другой файл с последующим его сжатием. Файл журнала при этом очищается. Ниже приводится короткий сценарий, осуществляющий такую процедуру для файла журнала */var/log/messages*:

```
mv /var/log/messages /var/log/messages`backup
cp /dev/null /var/log/messages

CURDATE=`date +"%m%d%y"`

mv /var/log/messages-backup /var/log/messages-$CURDATE
gzip /var/log/messages-$CURDATE
```

Прежде всего, мы копируем журнальный файл под другим именем и усекаем исходный файл до нулевой длины путем копирования в него из устройства */dev/null*. Это делается для того, чтобы можно было без помех продолжить ведение журнала, пока выполняются следующие команды сценария. Затем мы создаем строку из текущей даты, которая будет использоваться в качестве суффикса имени файла, переименовываем сохраняемую копию и, наконец, сжимаем ее с помощью *gzip*.

Возможно, вы пожелаете выполнять этот сценарий через *cron*, но в том виде, как он здесь представлен, его нельзя выполнять чаще, чем раз в сутки, иначе сжатая резервная копия будет перезаписана, так как в имени файла отражена только текущая дата, но не время суток. Если вы хотите выполнять этот сценарий чаще, следует добавить к имени другие цифры, чтобы копии различались между собой.

Можно сделать много других усовершенствований. Например, вы можете сначала проверить размер журнального файла и создавать его сжатую копию только в случае превышения заданного порога. Даже несмотря на такое усовершенствование, на вашем разделе, содержащем журнальные файлы, с течением времени не останется места. Эту проблему можно решить, установив максимальное количество (например, 10) сохраняемых сжатых файлов журналов. Когда число журнальных файлов достигает установленного вами предела, самый старый из них уничтожается, и поверх него записывается очередной копируемый. Такая схема называется ротацией журналов. В некоторых дистрибутивах есть сценарии типа *savelog* или *logrotate*, автоматически производящие такую операцию.

Завершая обсуждение этой темы, отметим, что в новейших дистрибутивах, например SUSE, Debian и Red Hat, уже есть встроенные сценарии управления файлами журналов для *cron*; они значительно сложнее представленного здесь маленького сценария.

Процессы

Понятие процесса является центральным в UNIX. Освоение этого понятия поможет вам как пользователю управлять своим сеансом работы. Если вы также являетесь системным администратором, это понятие еще более важно.

Процесс (*process*) – это независимо выполняющаяся программа со своими ресурсами. Например, выше мы показывали, как можно переназначить вывод про-

граммы в файл, в то время как оболочка продолжает вывод на экран. Оболочка и другая программа могут посылать выходные данные в разные места благодаря тому, что они представляют собой разные процессы.

В UNIX ограниченные ресурсы системы, такие как оперативная память и диски, управляются одной всемогущей программой, называемой ядром. Все остальное в системе есть процесс.

Таким образом, пока вы не вошли в систему, ваш терминал управляется процессом *getty*. После регистрации процесс *getty* «умирает» (при выходе из сеанса ядро запускает новый процесс), и ваш терминал переходит под управление командной оболочки, которая является совсем другим процессом. Оболочка создает новый процесс при каждом вводе команды. Создание нового процесса называется *ветвлением* (*forking*), поскольку один процесс расщепляется на два.

Если вы работаете в X Window System, то каждый процесс открывает одно или более окон. Так, владельцем окна, в котором вы вводите команды, является процесс *xterm*. Этот процесс разветвляет оболочку, чтобы она выполнялась в окне. А эта оболочка порождает дополнительные процессы, когда вы вводите команды.

Чтобы увидеть процессы, которые у вас выполняются, введите команду `ps`. На рис. 10.2 показаны типичные результаты ее работы и объяснено значение полей. Вас может удивить, как много процессов выполняется, особенно если вы работаете в X. Команда `ps` сама является одним из процессов, который, конечно, «умирает», как только вывод данных осуществлен.

Первым полем вывода *ps* является уникальный идентификатор процесса. Если у вас есть вышедший из-под контроля процесс, от которого нельзя избавиться с помощью комбинации Ctrl+C или другим способом, можно убить его, перейдя в другую виртуальную консоль или окно X и введя:

```
$ kill process-id
```

Поле `TTY` показывает, на каком терминале работает процесс, если таковой имеется. (Все, что запускается из оболочки, использует, конечно, терминал, но у демонов, запущенных в фоновом режиме, нет терминала.)

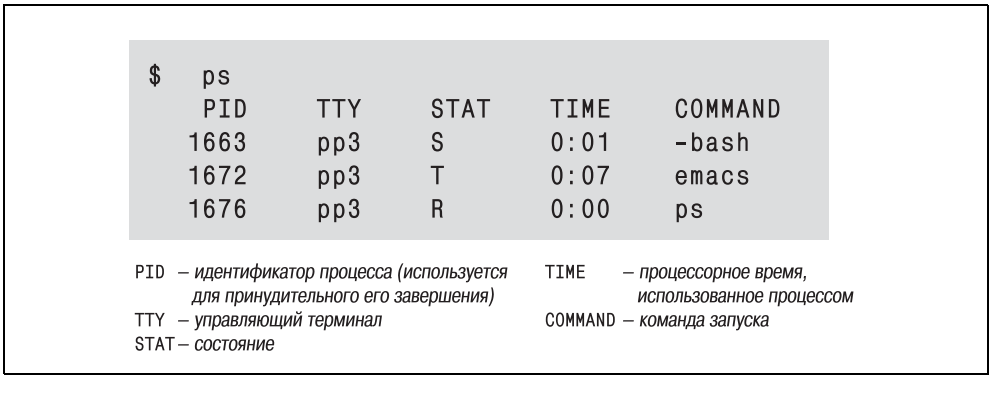


Рис. 10.2. *Результат работы команды ps*

Поле STAT показывает состояние процесса. Оболочка в данный момент находится в режиме ожидания (suspended), поэтому в этом поле стоит символ S. Выполняется сеанс редактирования в Emacs, но он приостановлен нажатием Ctrl+Z. Это отмечено знаком T в поле STAT. Последним показан процесс ps, создающий этот вывод. Его состояние, конечно, R (running), поскольку он выполняется.

Поле TIME показывает, сколько процессорного времени использовали процессы. Поскольку *bash* и Emacs являются интерактивными, они мало используют процессор.

Можно посмотреть не только свои процессы, но и все процессы системы. Взглянем на все процессы в системе. Параметр a задает все процессы, а параметр x позволяет включить процессы, у которых нет управляющего терминала (такие, как демоны, запущенные во время работы):

```
$ ps ax | more
```

Теперь вы можете увидеть демоны, о которых говорилось в предыдущем разделе.

В последних версиях команды ps есть приятный дополнительный параметр. Если вы ищете определенный процесс и знаете его имя или хотя бы часть его, можно воспользоваться параметром -C, указав за ним это имя, и увидеть только те процессы, имена которых соответствуют заданному:

```
$ ps -C httpd
```

И на этом месте, глядя на захватывающую дыхание картину всей системы UNIX за работой, мы заканчиваем обсуждение процессов (строки ограничены 76 колонками; если вы хотите увидеть их в полном блеске, добавьте к команде ps параметр -w):

```
kalle@owl:~ > ps aux
USER     PID %CPU %MEM    VSZ   RSS TTY    STAT START   TIME COMMAND
root       1  0.0  0.0    588   240 ?      S    14:49   0:05 init [3]
root       2  0.0  0.0      0     0 ?      S    14:49   0:00 [migration/0]
root       3  0.0  0.0      0     0 ?      SN   14:49   0:00 [ksoftirqd/0]
root       4  0.0  0.0      0     0 ?      S    14:49   0:00 [migration/1]
root       5  0.0  0.0      0     0 ?      SN   14:49   0:00 [ksoftirqd/1]
root       6  0.0  0.0      0     0 ?      S<   14:49   0:00 [events/0]
root       7  0.0  0.0      0     0 ?      S<   14:49   0:00 [events/1]
root       8  0.0  0.0      0     0 ?      S<   14:49   0:00 [kacpid]
root       9  0.0  0.0      0     0 ?      S<   14:49   0:00 [kblockd/0]
root      10  0.0  0.0      0     0 ?      S<   14:49   0:00 [kblockd/1]
root      11  0.0  0.0      0     0 ?      S    14:49   0:00 [kirqd]
root      14  0.0  0.0      0     0 ?      S<   14:49   0:00 [khelper]
root      15  0.0  0.0      0     0 ?      S    14:49   0:00 [pdflush]
root      16  0.0  0.0      0     0 ?      S    14:49   0:00 [pdflush]
root      17  0.0  0.0      0     0 ?      S    14:49   0:00 [kswapd0]
root      18  0.0  0.0      0     0 ?      S<   14:49   0:00 [aio/0]
root      19  0.0  0.0      0     0 ?      S<   14:49   0:00 [aio/1]
root     689  0.0  0.0      0     0 ?      S    14:49   0:00 [kseriod]
root     966  0.0  0.0      0     0 ?      S    14:49   0:00 [scsi_eh_0]
root    1138  0.0  0.0      0     0 ?      S    14:49   0:00 [kjournald]
root    1584  0.0  0.0      0     0 ?      S    14:49   0:00 [kjournald]
root    1585  0.0  0.0      0     0 ?      S    14:49   0:00 [kjournald]
```

```
root     1586  0.0  0.0      0     0 ?      S    14:49   0:00 [kjournald]
root     2466  0.0  0.0      0     0 ?      S    14:49   0:00 [khubd]
root     2958  0.0  0.0   1412   436 ?      S    14:49   0:00 [hwscand]
root     3759  0.0  0.0   1436   612 ?      Ss   14:49   0:00 /sbin/syslogd -a
root     3798  0.0  0.1   2352  1516 ?      Ss   14:49   0:00 /sbin/klogd -c 1
bin      3858  0.0  0.0   1420   492 ?      Ss   14:49   0:00 /sbin/portmap
root     3868  0.0  0.0   1588   652 ?      Ss   14:49   0:00 /sbin/resmgrd
root     3892  0.0  0.0   1396   544 ?      Ss   14:49   0:00 hcid: processing
root     3898  0.0  0.0   1420   528 ?      Ss   14:49   0:00 /usr/sbin/sdpd
root     4356  0.0  0.0      0     0 ?      S    14:49   0:00 [usb-storage]
root     4357  0.0  0.0      0     0 ?      S    14:49   0:00 [scsi_eh_1]
root     4368  0.0  0.1   4708  1804 ?      Ss   14:49   0:00 /usr/sbin/sshd -o
root     4715  0.0  0.1   2600  1240 ?      S    14:49   0:00 /usr/sbin/powersa
lp       4905  0.0  0.3   6416  3392 ?      Ss   14:49   0:00 /usr/sbin/cupsd
root     5103  0.0  0.1   4176  1432 ?      Ss   14:49   0:00 /usr/lib/postfix/
postfix  5193  0.0  0.1   4252  1512 ?      S    14:49   0:00 qmgr -l -t fifo -
root     5219  0.0  0.0   1584   704 ?      Ss   14:49   0:00 /usr/sbin/cron
root     5222  0.0  0.0  42624   784 ?      Ss   14:49   0:00 /usr/sbin/nscd
root     5537  0.0  0.1   2264  1216 ?      Ss   14:49   0:00 login -- kalle
root     5538  0.0  0.0   1608   608 tty2   Ss+  14:49   0:00 /sbin/mingetty tt
root     5539  0.0  0.0   1608   608 tty3   Ss+  14:49   0:00 /sbin/mingetty tt
root     5540  0.0  0.0   1608   608 tty4   Ss+  14:49   0:00 /sbin/mingetty tt
root     5541  0.0  0.0   1608   608 tty5   Ss+  14:49   0:00 /sbin/mingetty tt
root     5542  0.0  0.0   1608   608 tty6   Ss+  14:49   0:00 /sbin/mingetty tt
kalle    5556  0.0  0.1   4180  1996 tty1   Ss   14:50   0:00 -zsh
kalle    5572  0.0  0.0   3012   816 ?      Ss   14:50   0:00 gpg-agent --daemo
kalle    5574  0.0  0.1   4296  1332 ?      Ss   14:50   0:00 ssh-agent
kalle    5579  0.0  0.1   3708  1248 tty1   S+   14:50   0:00 /bin/sh /usr/X11R
kalle    5580  0.0  0.0   2504   564 tty1   S+   14:50   0:00 tee /home/kalle/.
kalle    5592  0.0  0.0   2384   652 tty1   S+   14:50   0:00 xinit /home/kalle
root     5593  3.4  4.5 106948 46744 ?      S    14:50   7:12 X :0 -auth /home/
kalle    5619  0.0  0.1   3704  1288 tty1   S    14:50   0:00 /bin/sh /usr/X11R
kalle    5658  0.0  1.0  24252 10412 ?      Ss   14:50   0:00 kdeinit Running..
kalle    5661  0.0  0.8  22876  8976 ?      S    14:50   0:00 kdeinit: dcopserv
kalle    5663  0.0  1.0  25340 10916 ?      S    14:50   0:00 kdeinit: klaunche
kalle    5666  0.0  1.7  31316 18540 ?      S    14:50   0:05 kdeinit: kded
kalle    5673  0.0  1.3  26480 14292 ?      S    14:50   0:00 kdeinit: kxkb
kalle    5677  0.0  0.5   9820  5736 ?      S    14:50   0:00 /opt/kde3/bin/art
kalle    5679  0.0  0.0   1372   336 tty1   S    14:50   0:00 kwrapper ksmserve
kalle    5681  0.0  1.1  24800 12116 ?      S    14:50   0:00 kdeinit: ksmserve
kalle    5683  0.0  1.4  27464 15512 ?      S    14:50   0:09 kdeinit: kwin -se
kalle    5686  0.0  1.8  30160 18920 ?      S    14:50   0:05 kdeinit: kdesktop
kalle    5688  0.1  1.8  31748 19460 ?      S    14:50   0:19 kdeinit: kicker
kalle    5689  0.0  1.0  25856 11360 ?      S    14:50   0:00 kdeinit: kio_file
kalle    5692  0.0  1.3  26324 14304 ?      S    14:50   0:02 kdeinit: klipper
kalle    5693  0.0  0.7  21144  7908 ?      S    14:50   0:00 kpowersave
kalle    5698  0.0  1.3  25840 13804 ?      S    14:50   0:00 kamix
kalle    5701  0.0  1.2  24764 12668 ?      S    14:50   0:00 kpowersave
kalle    5705  0.0  1.4  29260 15260 ?      S    14:50   0:01 suseplugger -capt
kalle    5706  0.0  1.2  24720 13376 ?      S    14:50   0:00 susewatcher -capt
kalle    5707  0.0  1.6  28476 16564 ?      S    14:50   0:00 kgpg
kalle    5713  0.0  1.2  25088 12468 ?      S    14:50   0:02 kdeinit: khotkeys
kalle    5715  0.0  1.9  30296 19920 ?      S    14:50   0:08 oooqs -caption Op
kalle    5717  0.0  1.5  28452 15716 ?      S    14:50   0:00 kdeinit: kio_uise
```

```
kalle    5740  0.0  1.0  26040 11260 ?      S    14:50   0:00 kdeinit: kio_file
kalle    5748  0.0  1.6  30084 16928 ?      S    14:50   0:05 kdeinit: konsole
kalle    5750  1.8  4.0  57404 42244 ?      S    14:50   3:48 kontact -session
kalle    5751  0.0  1.6  29968 16632 ?      S    14:50   0:00 kdeinit: konsole
kalle    5754  0.0  0.5  14968  5976 ?      S    14:50   0:00 /opt/kde3/bin/kde
kalle    5757  0.0  0.1   4188  1920 pts/2  Ss+  14:50   0:00 /bin/zsh
kalle    5759  0.0  0.1   4188  1944 pts/3  Ss   14:50   0:00 /bin/zsh
kalle    5761  0.0  0.2   4684  2572 pts/4  Ss+  14:50   0:00 /bin/zsh
kalle    5800  0.0  0.9  24484  9988 ?      S    14:50   0:00 kalarmd --login
kalle    5803  0.0  2.6  36264 27472 ?      S    14:50   0:05 xemacs
kalle    5826  0.0  0.1   3704  1172 pts/3  S+   14:51   0:00 sh ./sshtunnel
kalle    5827  0.0  0.2   4956  2348 pts/3  S+   14:51   0:02 ssh -X -L 23456:1
kalle    5829  0.1  1.9  31008 20204 ?      S    14:51   0:20 kdeinit: ksirc -i
kalle    6086  0.0  0.1   3444  1244 ?      S    15:07   0:00 /bin/sh /home/kal
kalle    6107  0.0  0.1   3704  1264 ?      S    15:07   0:00 /bin/sh /home/kal
kalle    6115  0.7  4.2  71184 43512 ?      S    15:07   1:29 /home/kalle/firef
kalle    6118  0.0  0.3   6460  3612 ?      S    15:07   0:00 /opt/gnome/lib/GC
kalle    6137  0.0  0.5   8232  5616 ?      S    15:08   0:03 perl /opt/kde3/bi
kalle    6186  0.0  2.9  42300 30384 ?      S    15:10   0:03 kdeinit: konquero
kalle    6497  0.1  1.6  30592 17424 ?      R    15:20   0:11 kdeinit: konsole
kalle    6498  0.0  0.2   4724  2624 pts/1  Ss+  15:20   0:00 /bin/zsh
kalle    6511  0.9  3.0  39932 31456 pts/1  S    15:20   1:37 xemacs
kalle    6720  0.0  0.2   4584  2500 pts/5  Ss   15:32   0:00 /bin/zsh
root     6740  0.0  0.1   3480  1264 pts/5  S    15:32   0:00 su
root     6741  0.0  0.1   3608  1732 pts/5  S    15:32   0:00 bash
kalle    6818  0.0  1.6  30152 17316 ?      S    15:39   0:00 kdeinit: konsole
kalle    6819  0.0  0.2   4492  2396 pts/6  Ss+  15:39   0:00 /bin/zsh
kalle    6948  0.0  1.6  29872 16564 ?      S    15:48   0:00 kdeinit: konsole
kalle    6949  0.0  0.1   4188  2040 pts/7  Ss   15:48   0:00 /bin/zsh
kalle    6982  0.0  0.1   4556  1908 pts/7  S+   15:50   0:00 ssh cvs.kdab.net
at       8106  0.0  0.0   1432   536 ?      Ss   17:24   0:00 /usr/sbin/atd
postfix  8672  0.0  0.1   4220  1448 ?      S    18:09   0:00 pickup -l -t fifo
postfix  8779  0.0  0.1   4208  1396 ?      S    18:15   0:00 proxymap -t unix
postfix  8796  0.0  0.1   4744  1784 ?      S    18:17   0:00 trivial-rewrite -
postfix  8797  0.0  0.1   4904  1848 ?      S    18:17   0:00 cleanup -z -t uni
postfix  8798  0.0  0.1   4376  1768 ?      S    18:17   0:00 local -t unix
root     8807  0.0  0.0   1584   700 ?      S    18:19   0:00 /USR/SBIN/CRON
kalle    8808  0.0  0.1   3112  1144 ?      Ss   18:19   0:00 fetchmail
root     8822  0.0  0.0   2164   688 pts/5  R+   18:20   0:00 ps aux
```

Обслуживающие программы

Мы включили в книгу этот раздел, поскольку вам пора заинтересоваться тем, что же выполняется в системе неведомо для вас.

В наше время компьютер обычно производит более сложную работу, чем просмотр файла или другого статического ресурса. Иногда эта работа требует взаимодействия между работающими процессами.

Возьмем, например, протокол FTP, которым вы, возможно, пользовались при загрузке относящихся к Linux документов и программ. Когда вы подключаетесь по FTP к другой системе, на ней должна работать программа, принимающая ваше подключение и интерпретирующая ваши команды. Поэтому на той системе

работает некоторая программа по имени *ftpd*. Буква *d* в имени происходит от *daemon* (*демон*), что является причудливым термином UNIX для обозначения сервера, постоянно работающего в фоновом режиме. Большинство демонов заняты поддержкой работы в сети.

Вы, наверное, слишком часто слышали навязшее в зубах выражение «клиент-сервер», но это как раз то, что здесь применяется, более того, применяется десятки лет.

Демоны запускаются при загрузке системы. Чтобы увидеть, как они запускаются, просмотрите файлы */etc/inittab* и */etc/inetd.conf* и специфические для дистрибутива файлы с настройками. Их формат мы сейчас не будем обсуждать. Но в каждой строке этих файлов указана программа, запускаемая при старте системы. Специфические для дистрибутива файлы можно найти, просмотрев поставляемую с системой документацию или обратив внимание на имена путей, чаще всего встречающихся в */etc/inittab*. Обычно они указывают на дерево каталогов, в которых ваш дистрибутив хранит стартовые файлы системы.

Чтобы увидеть пример того, как ваша система использует */etc/inittab*, взгляните на строки, содержащие `getty` или `agetty`. Это программа, которая прослушивает терминал (tty) и ждет регистрации пользователя. В ней выводится приглашение `login:`, о котором мы говорили в начале главы.

Файл */etc/inetd.conf* представляет собой более сложный способ запуска программ – еще один уровень перенаправления. Идея использования */etc/inetd.conf* состоит в том, что было бы напрасной тратой ресурсов системы, если бы десяток или более демонов крутились без дела, ожидая, пока поступит запрос из сети. Вместо этого система запускает один демон с именем *inetd*. Этот демон ждет поступления запросов на соединение от клиентов на других машинах и при осуществлении входящего соединения запускает для его обработки соответствующий демон. Например, при осуществлении входящего FTP-соединения *inetd* запускает FTP-демон (*ftpd*) для управления соединением. Благодаря этому выполняются только те сетевые демоны, которые действительно используются.

Для каждого сервиса, предлагаемого системой другим машинам в сети, есть свой демон: *fingerd* – для обработки удаленных запросов *finger*, *rwhod* – для обработки запросов *rwho*, и т. д. Некоторые демоны поддерживают несетевые сервисы, например, *kerneld* занимается автоматической загрузкой модулей в ядро. (В версиях 2.4 и выше эту задачу решает `kmod`, который теперь к тому же является не отдельным процессом, а потоком исполнения ядра.)

11

Управление пользователями, группами и привилегиями

Управление учетными записями пользователей

Понимание того, как управлять учетными записями пользователей, необходимо, даже если вы единственное человеческое существо, использующее вашу Linux-систему, а если пользователей несколько, то тем более.

Учетные записи пользователей служат ряду целей в UNIX-системах. В первую очередь они позволяют различать пользователей системы с целью идентификации и обеспечения безопасности. У каждого пользователя есть личная учетная запись с отдельным именем пользователя и паролем. Как будет обсуждаться в разделе «Владение файлами и права доступа» далее в этой главе, пользователи могут устанавливать права доступа для своих файлов, разрешая или ограничивая доступ к ним со стороны других пользователей. Каждым файлом системы «владеет» конкретный пользователь, который может установить для него права доступа. Учетные записи пользователей служат для авторизации доступа к системе: доступ к машине могут получить только те, у кого есть учетные записи. Кроме того, учетные записи служат для идентификации пользователей, ведения системных журналов, указания отправителя в сообщениях электронной почты и т. д.

Помимо личных учетных записей в системе есть пользователи, обеспечивающие административные функции. Как мы видели, администратор системы использует учетную запись *root* для осуществления сопровождения системы, но обычно не для личного пользования ею. Доступ к таким учетным записям осуществляется с помощью команды *su*, позволяющей обратиться к другой учетной записи после регистрации через личную учетную запись.

Некоторые учетные записи в системе могут быть вовсе не предназначены для взаимодействия с пользователем. Такие учетные записи обычно используются системными демонами, которые должны иметь доступ к файлам системы через ID пользователя, отличного от *root* или личных учетных записей пользователей. Например, если вы настраиваете систему для получения новостей из телеконференций с другого сервера, демон новостей должен записывать статьи телеконференций в каталог спулинга, доступ к которому может иметь каждый, но только

пользователь-демон новостей может осуществлять в него запись. С учетной записью *news* не ассоциируется никакая личность, это «мнимый» пользователь, созданный только для демона новостей.

Одним из битов разрешения, которые можно установить на выполняемые файлы, является бит *setuid*, заставляющий программу выполняться с правами владельца файла. Например, если владельцем демона новостей является пользователь *news* и на выполняемом файле установлен бит *setuid*, демон работает так, как если бы был запущен пользователем *news*. Таким образом, демон получит право записи в каталог спулинга новостей, а все остальные пользователи будут иметь право чтения имеющихся там статей. Это особенность системы защиты. Программы чтения новостей предоставят пользователям лишь необходимые права в отношении каталога спулинга новостей, и никому не будет позволено безнаказанно резвиться в нем.

В качестве системного администратора вы должны создавать на своей машине учетные записи всех пользователей, реальных и виртуальных, и управлять ими. В большинстве случаев это безболезненная и легкая задача, но важно понять, как она решается.

Файл passwd

Для каждой учетной записи в системе есть запись в файле */etc/passwd*. Этот файл содержит по одной строке на каждого пользователя, в которой указан ряд атрибутов учетной записи, таких как имя пользователя, его настоящее имя и т. д.

Каждая запись в этом файле имеет следующий формат:

```
username:password:uid:gid:gecos:homedir:shell
```

Ниже объясняется значение этих полей:

username

Уникальная строка символов, идентифицирующая учетную запись. Для личных учетных записей это имя, под которым пользователь регистрируется в системе. В большинстве систем оно ограничено восемью алфавитно-цифровыми символами, например *larry* или *kirsten*.

password

Зашифрованное представление пароля пользователя. Это поле устанавливается программой `passwd`; используется однонаправленное шифрование, которое трудно (но можно) раскрыть. Его не надо устанавливать вручную, это делает за вас программа `passwd`. Обратите внимание, что если первый символ в поле *password* – звездочка (`*`), то учетная запись «отключена»; система не позволит этому пользователю регистрироваться. Подробнее об этом рассказывается в разделе «Создание учетных записей» далее в этой главе.

uid

Идентификатор пользователя (ID), уникальное целое число, присваиваемое системой учетной записи. Система использует поле *uid* внутри себя для задач, связанных с правами доступа к процессам и файлам. С целыми числами обращаться проще и удобнее, чем со строками символов. Таким образом, каждая учетная запись идентифицируется как *uid*, так и именем пользователя *user-*

name: *uid* для системы более важен, в то время как имя пользователя удобнее для людей.

gid

ID группы, целое число, указывающее на группу пользователя по умолчанию, находящуюся в файле */etc/group*. Подробнее об этом см. в разделе «Файл group» далее в этой главе.

gecos

Различные сведения о пользователе, в том числе его настоящее имя, необязательные адресные сведения типа служебного адреса или номера телефона. Эти данные используют такие программы, как *mail* и *finger*, для идентификации пользователей в системе; мы еще поговорим об этом позднее. Между прочим, название *gecos* – историческое (появилось в 1970-е годы) и происходит от *General Electric Comprehensive Operating System*. GECOS не имеет никакого отношения к UNIX, кроме того, что первоначально это поле было добавлено к */etc/passwd* для обеспечения совместимости с некоторыми ее сервисами.

homedir

Домашний каталог пользователя, предназначенный для личного использования; подробнее о нем рассказывается далее. При входе пользователя в систему оболочка считает текущим рабочим каталогом указанный домашний каталог.

shell

Имя программы, запускаемой при регистрации пользователя. Как правило, это полный путь к оболочке, например */bin/bash* или */bin/tcsh*.

Многие поля являются необязательными; обязательными полями являются только *username*, *uid*, *gid* и *homedir*. В большей части учетных записей заполнены все поля, однако «мнимые» или административные учетные записи могут использовать только некоторые.

Ниже приводится пример двух записей в */etc/passwd*:

```
root:ZxPsI9ZjiVd9Y:0:0:The root of all evil:/root:/bin/bash
aclark:BjDf5hBysDsii:104:50:Anna Clark:/home/aclark:/bin/bash
```

Первая запись относится к учетной записи *root*. Прежде всего, обратите внимание на то, что идентификатор пользователя *root* равен нулю. Благодаря этому *root* является суперпользователем: система знает, что *uid*, равный 0, является «особым» и у него нет обычных ограничений прав доступа. *gid* для *root* является также нулем, что является обычным соглашением. Владельцем многих файлов системы являются пользователь *root* и группа *root*, для которых *uid* и *gid* равны нулю. Подробнее о группах чуть позже.

Во многих системах домашним каталогом пользователя *root* является */root* или просто /. Обычно это не важно, поскольку чаще всего вы используете `su` для обретения привилегий пользователя *root* из своей учетной записи. Традиционным является также использование какого-нибудь варианта командной оболочки Борна (в данном случае */bin/bash*) в качестве командной оболочки для учетной записи *root*, хотя при желании можно воспользоваться командной оболочкой C. (Командные оболочки обсуждались в главе 4.) Однако будьте осторожны: у оболочек Борна и C разный синтаксис, и переключение между ними при использовании привилегий *root* может привести к неясностям и ошибкам.

Вторая строка относится к реальному пользователю-человеку с именем пользователя *aclark*. В нашем случае *uid* равен 104. Технически поле *uid* может быть любым целым числом. Во многих системах учетные записи пользователей получают номера от 100 и выше, а административные учетные записи – в диапазоне до 100. Идентификатор группы gid равен 50, это означает, что пользователь *aclark* принадлежит группе, имеющей номер 50 в файле */etc/group*. О группах будет рассказано в разделе «Файл group» далее в этой главе.

Домашние каталоги часто находятся в каталоге */home* и называются по имени владельца. Как правило, это удобное соглашение, позволяющее избежать путаницы при поиске домашнего каталога конкретного пользователя, но технически домашний каталог можно расположить в любом месте. Следует, однако, придерживаться структуры каталогов, принятой в вашей системе.

В качестве системного администратора вам вряд ли придется непосредственно редактировать файл */etc/passwd*. Есть несколько программ, позволяющих создавать и поддерживать учетные записи пользователей; подробнее об этом речь пойдет в разделе «Создание учетных записей» далее в этой главе. Если же вам действительно потребуется напрямую редактировать файл */etc/passwd*, вам пригодится команда `vipw`, которая защищает файл паролей от повреждения при одновременном редактировании несколькими администраторами.

Теневые пароли

Возможность видеть зашифрованные пароли в файле */etc/passwd*, которая есть у всех имеющих доступ к системе, создает некоторую угрозу безопасности. Есть специальные программы для взлома, перебирающие огромное число возможных паролей и сравнивающие результат их шифрования с заданным паролем.

Для преодоления такой потенциальной угрозы безопасности были придуманы *теневые пароли* (*shadow passwords*). При использовании теневых паролей поле *password* в файле */etc/passwd* содержит лишь символы `x` или `*`, чего никогда не может быть в зашифрованном варианте пароля. Для хранения паролей используется второй файл с именем */etc/shadow*. В нем содержатся записи, очень похожие на записи в */etc/passwd*, но содержащие реальные зашифрованные пароли в поле *password*. Файл */etc/shadow* доступен на чтение только пользователю, обладающему привилегиями суперпользователя, поэтому у обычных пользователей нет доступа к зашифрованным паролям. Остальные поля, кроме *username* и *password*, тоже есть в */etc/shadow*, но обычно ничего не содержат или содержат фиктивные значения.

Учтите, что для использования теневых паролей нужны специальные версии программ, имеющие доступ к данным о пользователях, таким как `passwd` или `login`, и возможность модифицировать эти данные. Сейчас большинство дистрибутивов поставляются с уже установленными теневыми паролями, так что это не должно вызвать проблем. Пользователи Debian должны задать параметр `shadowconfig on`, чтобы включить поддержку теневых паролей.

Существуют два инструментальных средства для преобразования «нормальных» учетных записей в теневые и обратно. Утилита `pwconv` берет файл */etc/passwd*, ищет в нем записи, которых еще нет в */etc/shadow*, создает для них теневые записи и объединяет с уже имеющимися в файле */etc/shadow*.

Утилита `pwunconv` используется редко, поскольку не усиливает защиту, а ослабляет ее. Она работает подобно `pwconv`, но создает обычные записи в */etc/passwd*, которые могут работать без соответствующих им записей в */etc/shadow*.

В современных системах Linux помимо теневых паролей существует возможность ограничения срока действия пароля. За несколько дней до истечения срока действия пароля (определяется настройками) пользователь получит предупреждение и предложение изменить пароль. Если этого сделано не будет, через некоторое время его учетная запись окажется заблокированной. Кроме того, имеется возможность установить минимальное число дней, прежде чем измененный или вновь созданный пароль может быть изменен снова.

Все эти параметры настраиваются с помощью команды `passwd`. Параметр *-n* устанавливает минимальное число дней между изменениями, *-x* – максимальное число дней между изменениями, *-w* – число дней, оставшихся до истечения срока действия пароля, когда будет выдано предупреждение, и *-i* – число дней между моментом истечения срока действия пароля и моментом, когда учетная запись будет заблокирована.

В большинстве дистрибутивов имеются программы управления всеми этими настройками с помощью графического интерфейса, которые нередко находятся на странице с названием «Дополнительные настройки» или похожим.

PAM и другие способы аутентификации

Если вы решили, что двух способов аутентификации – */etc/passwd* и */etc/shadow* – достаточно для выбора, то вы ошибаетесь. Есть ряд других методов аутентификации с необычными именами, такими как аутентификация Kerberos (по имени собаки в греческой мифологии, охранявшей вход в ад). Хотя мы считаем, что теневые пароли в большинстве случаев предоставляют достаточную защиту, все зависит от того, какая степень защиты вам действительно нужна и в какой степени вы страдаете паранойей.

Проблема с этими методами аутентификации состоит в том, что нет простого способа переключения с одного метода на другой, поскольку вместе с этими средствами всегда должен устанавливаться набор программ типа `login` и `passwd`. Для решения этой проблемы придумана система *подключаемых методов аутентификации* (*Pluggable Authentification Methods*, *PAM*). Если у вас есть набор средств, поддерживающих PAM, вы можете изменять используемый в системе метод аутентификации путем перенастройки PAM. Эти средства автоматически получают программный код, необходимый для требуемых процедур аутентификации, из динамически загружаемых библиотек совместного доступа.

Установка и применение PAM выходят за рамки данной книги, но необходимые сведения вы можете получить на *http://www.kernel.org/pub/linux/libs/pam/*. Большинство современных дистрибутивов устанавливают PAM.

Файл group

Группы пользователей являются удобным способом логической организации наборов учетных записей пользователей и предоставляют пользователям возможность иметь общий доступ к файлам внутри одной или нескольких групп. У каж-

дого файла в системе есть пользователь и группа, которые им владеют. Для каждого конкретного файла вы можете посмотреть владельца и группу с помощью команды *ls -l*, например:

```
rutabaga% ls -l boiler.tex
-rwxrw-r--   1 mdw      megabozo    10316 Oct   6 20:19 boiler.tex
rutabaga%
```

Владельцем этого файла является пользователь *mdw*, и принадлежит он группе *megabozo*. Права доступа говорят о том, что пользователь *mdw* имеет право на чтение, запись и выполнение файла; все члены группы *megabozo* имеют право на чтение и на запись; все остальные пользователи имеют право только на чтение.

Это не означает, что пользователь *mdw* является членом группы *megabozo,* просто доступ к файлу согласно битам разрешений предоставлен всем членам группы *megabozo* (в состав которой может входить или не входить пользователь *mdw*).

Таким образом, организован совместный доступ к файлам групп пользователей, а права доступа могут отдельно указываться для владельца, группы, к которой относится файл, и всех остальных. С правами доступа можно ознакомиться в разделе «Владение файлами и права доступа» далее в этой главе.

Каждый пользователь принадлежит по крайней мере к одной группе, указанной в поле *gid* файла */etc/passwd*. Однако пользователь может быть членом нескольких групп. В файле */etc/group* содержится по одной строке для каждой группы в системе, что очень похоже на */etc/passwd*. Формат этого файла следующий:

```
groupname:password:gid:members
```

Здесь *groupname* является строкой символов, идентифицирующей группу; это то имя группы, которое выводится такими командами, как *ls -l*.

password является необязательным паролем, присвоенным группе, который дает возможность пользователям, не принадлежащим этой группе, присоединяться к группе с помощью команды `newgrp`. Об этом будет рассказано ниже.

gid является числовым идентификатором (ID) группы, используемым системой при ссылке на группу; это же число находится в поле *gid* файла */etc/passwd* для указания группы пользователя по умолчанию.

members – это список имен пользователей, разделенных запятыми (без пробелов между ними), указывающий пользователей, являющихся членами этой группы, но имеющих другие значения *gid* в файле */etc/passwd*. Это означает, что в данный список не нужно включать имена тех пользователей, для которых эта группа указана как группа по умолчанию в */etc/passwd*, то есть это список дополнительных членов группы.

Например, */etc/group* может содержать такие записи:

```
root:*:0:
bin:*:1:root,daemon
users:*:50:
bozo:*:51:linus,mdw
megabozo:*:52:kibo
```

Первые записи, для групп *root* и *bin*, являются административными группами, сходными, по сути, с «мнимыми» учетными записями в системе. Владельцами

многих файлов являются группы, такие как *root* и *bin*. Другие группы предназначены для учетных записей пользователей. Подобно числовым идентификаторам пользователей, ID групп пользователей часто принимают значения, большие 50 или 100.

Поле *password* файла групп представляет известный интерес. Используется оно нечасто, но вместе с программой *newgrp* позволяет пользователям, не являющимся членами данной группы, принять ID этой группы при условии знания пароля. Например, команда

```
rutabaga% newgrp bozo
Password: пароль для группы bozo
rutabaga%
```

запускает новую оболочку с ID группы `bozo`. Если поле *password* пустое или первым символом является звездочка, то команда *newgrp* отказывает в доступе к этой группе.

Однако поле *password* файла групп используется редко и на самом деле не является необходимым. (Фактически большинство систем не имеют средств установки пароля для группы; можно воспользоваться `passwd`, чтобы установить пароль для фиктивного пользователя с именем, совпадающим с именем группы, в файле */etc/passwd* и скопировать зашифрованное поле пароля в файл */etc/group*.) Вместо этого можно сделать пользователя членом нескольких групп, просто включив имя пользователя в поле *members* для каждой дополнительной группы. В предыдущем примере пользователи *linus* и *mdw* являются членами группы *bozo* и той группы, к которой они приписаны в файле */etc/passwd*. Если мы хотим добавить пользователя *linus* еще и к группе *megabozo*, последнюю строку предыдущего примера нужно заменить такой:

```
megabozo:*:52:kibo,linus
```

Команда `groups` выводит список групп, к которым вы принадлежите, например:

```
rutabaga% groups
users bozo
```

Если передать команде `groups` список пользователей, она выведет списки групп, к которым принадлежит каждый из пользователей.

При входе в систему вам автоматически присваиваются ID группы из */etc/passwd*, а также дополнительных групп, в списки которых вы включены в файле */etc/group*. Это значит, что у вас есть «групповой доступ» ко всем файлам в системе, для которых ID группы содержится в вашем списке групп. В этом случае биты прав доступа к этому файлу для группы (установленные командой *chmod g+...*) применимы к вам. (Если только вы не являетесь его владельцем, когда применяются биты прав доступа владельца.)

Теперь вам все известно о группах. Как же следует пользоваться возможностью создания групп? Это дело вкуса и зависит от того, как будет использоваться ваша система. Если пользователь в системе один или их мало, проще создать единственную группу, скажем, *users*, к которой принадлежат все учетные записи. Заметьте, что все системные группы, содержащиеся в файле */etc/group* после первоначальной установки, скорее всего, следует сохранить: от них могут зависеть различные демоны и программы.

Если пользователей на машине много, то есть несколько способов организовать их в группы. Например, в учебном заведении может быть несколько групп для студентов, кафедр и персонала. В компании, разрабатывающей программное обеспечение, для каждой бригады разработчиков может быть создана группа. В некоторых случаях каждого пользователя помещают в отдельную группу с именем, совпадающим с именем пользователя. В результате каждый живет сам по себе. Для файлов можно создавать специальные группы. Многие пользователи создают новые группы и делают их владельцами файлов, чтобы обеспечить совместный доступ пользователей к этим файлам. Однако при этом нужно вводить пользователей в эти новые группы, для чего обычно требуется вмешательство администратора (для редактирования файла */etc/group* или действий с утилитой `gpasswd` в дистрибутиве Debian). Решайте сами, как поступить.

Другая ситуация, в которой часто используются группы, – это разделение по правам доступа к аппаратным устройствам. Допустим, что у вас есть сканер, доступ к которому осуществляется через */dev/scanner*. Если вы не хотите, чтобы доступ к сканеру имели все, создайте группу с именем *scanner*, разрешите этой группе доступ к файлу */dev/scanner*, сделав его доступным для чтения этой группе и недоступным для всех остальных, и добавьте всех, кому разрешено пользоваться сканером, в группу *scanner* в файле */etc/groups*.

Создание учетных записей

Создание учетной записи пользователя производится в несколько этапов: добавление записи в */etc/passwd*, создание домашнего каталога пользователя и установка в нем файлов с настройками пользователя по умолчанию (например, *.bashrc*). К счастью, не нужно делать это вручную – почти во всех системах Linux есть программа *adduser*, которая сделает это за вас. В некоторых дистрибутивах, таких как Red Hat и SUSE, используется несколько иной набор инструментальных средств для создания и удаления учетных записей. Если примеры, приводимые в этом разделе, у вас работать не будут, обращайтесь к документации, которая поставляется в составе дистрибутива. (В дистрибутиве Red Hat имеется возможность управления учетными записями с помощью инструмента, называемого *control-panel*, а в SUSE – с помощью *yast2*; дистрибутив Debian включает в себя сценарий *adduser*, в некоторых версиях интерактивный, автоматически создающий пользователей на основе файла с настройками */etc/adduser.conf*.) Кроме того, есть графические программы управления пользователями, такие как *KUser* в KDE или System Tools в GNOME.

Работа с *adduser* от имени *root* выглядит так. В ответ на приглашение вы вводите запрашиваемые данные. Часто приглашение предлагает разумное значение по умолчанию, которое можно выбрать, нажав Enter:

```
Adding a new user. The username should not exceed 8 characters
in length, or you many run into problems later.
(Добавление нового пользователя. Имя пользователя не должно превышать
в длину 8 символов, иначе вы можете столкнуться с трудностями.)

Enter login name for new account (^C to quit): norbert
(Введите имя регистрации для новой учетной записи)

Editing information for new user [norbert]
```

```
(Редактирование сведений о новом пользователе norbert)

Full Name: Norbert Ebersol
(Полное имя)
GID [100]: 117
Checking for an available UID after 500
First unused uid is 501
(Проверка свободных UID, больших 500
Первый неиспользуемый UID - 501)

UID [501]: (enter)
Home Directory [/home/norbert]: (enter)
Shell [/bin/bash]: (enter)
Password [norbert]: (пароль пользователя norbert)

Information for new user [norbert]:
Home directory: [/home/norbert] Shell: [/bin/bash]
Password: [(пароль пользователя norbert)] uid: [501] gid: [51]

Is this correct? [y/N]: y
(Верно ли это?)

Adding login [norbert] and making directory [/home/norbert]
Adding the files from the /etc/skel directory:
(Добавление учетной записи и создание каталога /home/norbert
Добавление файлов из каталога /etc/skel:)
./.emacs -> /home/norbert/./.emacs
./.kermrc -> /home/norbert/./.kermrc
./.bashrc -> /home/norbert/./.bashrc
... more files ...
```

Здесь нет ничего необычного, просто вводите запрашиваемые данные или предлагаемые значения по умолчанию. Обратите внимание, что *adduser* использует 100 в качестве ID группы по умолчанию и ищет первый неиспользованный ID пользователя после 500 (500 используется как минимальное значение в SUSE и Red Hat, в Debian используется 1000). Принятие этих значений не должно вызвать конфликтов; в предыдущем примере мы ввели ID группы 117, поскольку решили, что это будет группа пользователя, и предложенный системой ID пользователя 501.

После создания учетной записи в исходный каталог пользователя копируются файлы из каталога */etc/skel*. В */etc/skel* содержатся «скелеты» файлов для новой учетной записи, представляющие собой файлы с настройками по умолчанию для нового пользователя (такие, как *.emacs* и *.bashrc*). Можно поместить сюда и другие файлы, если они нужны для учетных записей новых пользователей.[1]

После того как все это сделано, новая учетная запись готова к использованию и пользователь *norbert* может входить в систему, используя пароль, заданный при работе *adduser*. В целях безопасности новый пользователь должен сменить пароль сразу после первого входа в систему, используя утилиту *passwd*.

---

[1] Кроме того, там находится «заготовка» пустого каталога */tmp*, что очень важно: в скопированный отсюда дубликат */tmp* будут записываться все временные файлы пользователя во время работы разных программ. – *Примеч. науч. ред.*

root имеет возможность установить пароль для любого пользователя системы. Например, команда:

```
passwd norbert
```

приглашает ввести новый пароль для пользователя *norbert*, не спрашивая при этом старый пароль. Имейте, конечно, в виду, что нужно знать пароль суперпользователя, чтобы поменять его. Если вы забудете пароль для *root*, можно загрузить Linux с аварийной дискеты и очистить в */etc/passwd* поле *password* записи для *root*. Подробнее об этом см. в разделе «Действия в аварийных ситуациях» главы 27.

В некоторых системах Linux вместо *adduser* есть управляемая из командной строки утилита *useradd*. (Самое неприятное, что в некоторых системах эти две команды являются синонимами.) Она требует задания всей необходимой информации в виде параметров командной строки. Если вы не можете найти *adduser* и завязли с *useradd*, обращайтесь за помощью к страницам справочного руководства.

Удаление и блокировка учетных записей

Удалить учетную запись пользователя значительно легче, чем создать. Это хорошо известное понятие энтропии в действии. Для удаления записи нужно удалить запись о пользователе из */etc/passwd*, убрать все ссылки на пользователя из */etc/group* и удалить домашний каталог пользователя, а также другие файлы, владельцем или создателем которых пользователь является. Например, если у пользователя есть ящик для входящей электронной почты в */var/spool/mail*, его тоже нужно удалить.

Команда *userdel* (инь для ян – *useradd*) удаляет учетную запись и домашний каталог пользователя. Например, команда:

```
userdel -r norbert
```

удалит недавно созданную учетную запись для пользователя *norbert*. Параметр *-r* говорит о том, что надо удалить также домашний каталог. Прочие связанные с пользователем файлы, например почтовый ящик, файлы *crontab*, должны быть удалены вручную. Обычно они не важны и могут быть оставлены. К концу главы вы будете знать, где они находятся, если существуют. Быстро найти файлы, связанные с некоторым пользователем, можно с помощью команды:

```
find / -user username -ls
```

Эта команда выведет список файлов командой *ls -l*, владельцем которых является *username*. Разумеется, для ее использования необходимо, чтобы учетная запись *username* все еще присутствовала в */etc/passwd*. Если вы уже удалили учетную запись, используйте параметр вида *-uid num*, где *num* – числовой ID пользователя, почившего в бозе.

Еще проще при необходимости отключить запись пользователя на короткое (или не очень короткое) время. Можно удалить запись пользователя из */etc/passwd* (оставив нетронутыми домашний каталог и прочие его файлы) или добавить звездочку в качестве первого символа поля *password* записи в */etc/passwd*, например:

```
aclark:*BjDf5hBysDsii:104:50:Anna Clark:/home/aclark:/bin/bash
```

Тем самым будет блокирована регистрация по этой учетной записи. Если используются теневые пароли, то же самое нужно сделать в */etc/shadow*. В каких случаях это может пригодиться? Представьте себе, что служащий увольняется из компании и возникает необходимость исключить возможность входа этого пользователя в систему, но файлы, принадлежавшие этому пользователю, необходимо сохранить на тот случай, если они потребуются его коллегам. В подобных ситуациях бывает удобно просто отключить учетную запись, не удаляя домашний каталог (и другие файлы, связанные с именем пользователя, как, например, почтовый ящик).

Изменение учетных записей пользователей

Для изменения атрибутов учетных записей пользователей и групп обычно достаточно отредактировать записи в файлах */etc/passwd* и */etc/group*. Во многих системах для этой цели есть такие команды, как `usermod` и `groupmod`. Обычно оказывается проще отредактировать файлы вручную.

Для изменения пароля пользователя используется команда `passwd`, которая попросит ввести пароль, зашифрует его и запишет зашифрованный пароль в файл */etc/passwd*.

Если нужно изменить ID пользователя для существующей учетной записи, это можно сделать, непосредственно отредактировав поле `uid` в файле */etc/passwd*. Однако потребуется также изменить владельца файлов для этого пользователя на новый uid. Это можно сделать так:

```
chown -R aclark /home/aclark
```

При этом владельцем всех файлов в исходном каталоге, используемом *aclark*, снова станет *aclark*, если вы изменили uid для этой учетной записи. Если *ls -l* вместо имени пользователя выводит числовой ID, это значит, что нет имени пользователя, связанного с uid, владеющим файлами. Исправить положение можно с помощью команды `chown`.

Владение файлами и права доступа

Понятия «владелец файла» и «права доступа» – центральные в вопросах обеспечения безопасности. Их надо правильно устанавливать, даже если вы единственный пользователь, иначе могут произойти неприятности. С файлами, которые пользователи создают и используют ежедневно, эти атрибуты обычно работают, не требуя особых размышлений (хотя все же полезно иметь о них понятие). При администрировании системы все не так просто. Стоит назначить неверные права владения и доступа, и можно оказаться в сложной ситуации, не будучи, к примеру, в состоянии прочесть собственную почту. Как правило, сообщение

```
Permission denied
```

означает, что кто-то назначил права владения и доступа, ограничивающие доступ больше, чем нужно.

Права доступа

Права доступа (permissions) означают способ, которым кто-либо может использовать файл. В UNIX есть три разновидности прав доступа:

- Право на *чтение* (*read*) означает разрешение просмотреть содержимое файла.
- Право на *запись* (*write*) означает разрешение изменить или удалить файл.
- Право на *исполнение* (*execute*) означает разрешение выполнять файл как программу.

При создании каждого файла система назначает некоторые права по умолчанию, которых достаточно в большинстве случаев. Например, вы получаете права на чтение и запись, но остальные получают только право на чтение. Если вы одержимы паранойей, то можете сделать так, что у остальных не будет вообще никаких прав.

Кроме того, большинство утилит умеют устанавливать права доступа. Например, компилятор при создании исполняемой программы автоматически устанавливает для нее право на исполнение.

Однако бывают случаи, когда установок по умолчанию недостаточно. Например, если вы создаете сценарий оболочки или программу на языке программирования Perl, то должны присвоить себе право на их выполнение, иначе вы не сможете их запустить. Далее в этом разделе мы покажем, как это делать, но сначала разберемся с основными понятиями.

Для каталога те же права доступа имеют другой смысл:

- Право на чтение означает возможность вывода содержимого каталога.
- Право на запись означает возможность добавления и удаления файлов в этом каталоге.
- Право на исполнение означает разрешение вывода сведений о файлах в этом каталоге.

Не ищите разницу между правом на чтение и на выполнение для каталога. Обычно они идут в связке: присвойте оба или не присваивайте ни одного.

Обратите внимание, что, дав кому-либо право добавлять файл в каталог, вы тем самым даете ему и право удалять оттуда файлы. Эти привилегии появляются совместно при присвоении права записи. Есть, однако, способ дать пользователям право совместного доступа к каталогу и помешать им удалять файлы друг друга (см. раздел «Обновление другого программного обеспечения» в главе 12).

В UNIX помимо каталогов и обычных файлов, о которых мы до сих пор говорили, есть и другие файлы. Это специальные файлы (устройства), сокеты, символические ссылки и другие, причем у каждого типа существуют свои правила в отношении прав доступа. Но знание каждого типа в подробностях от вас не требуется.

Владельцы и группы

А кто же получает эти права доступа? Чтобы дать возможность пользователям совместно работать, в UNIX есть три уровня прав доступа: владелец, группа и все остальные. Понятие «остальные» охватывает всех, у кого есть доступ к системе, но кто не является владельцем или членом группы.

Идея создания групп состоит в том, чтобы обеспечить некоторому числу пользователей, например группе программистов, доступ к файлу. Скажем, программист, создающий исходный программный код, может оставить за собой право на запись, но остальным членам своей группы дать право на чтение, используя разрешение для группы. Что касается «остальных», то у них может вообще не быть прав. (Вы уверены, что ваш программный код этого достоин?)

У каждого файла есть владелец и группа. Обычно владельцем является тот, кто создал файл. Каждый пользователь принадлежит также к некоторой группе по умолчанию, и эта группа назначается каждому файлу, создаваемому пользователем. Можно, однако, создать много групп и приписать каждого пользователя к нескольким группам. Изменяя группу, назначенную файлу, можно дать доступ любой совокупности пользователей по вашему желанию. О группах мы уже говорили в разделе «Файл group».

Теперь у нас есть все элементы системы безопасности: три права доступа (на чтение, на запись и на исполнение) и три уровня (пользователь, группа, остальные). Посмотрим на некоторые типичные файлы с точки зрения прав доступа.

На рис. 11.1 показана типичная исполняемая программа. Мы получили этот вывод, выполнив команду *ls* с параметром *-l*.

Два полезных факта сразу обращают на себя внимание: владелец файла является автором данной книги и вашим верным спутником, *mdw*, в то время как группой является *lib* (возможно, группа, созданная для программистов, работающих над библиотеками). Но главная информация о правах доступа зашифрована в виде набора букв в левой части экрана.

Первый символ – дефис – указывает, что это обычный файл. Следующие три позиции относятся к владельцу; как можно было ожидать, у *mdw* есть все три разрешения. Очередные три позиции относятся к членам группы: они могут читать файл (не очень полезное свойство для двоичного файла) и выполнять его, но не могут записывать в него, поскольку в поле, которое должно содержать `w`, помещен дефис. И последние три поля относятся ко «всем остальным»; в данном случае у них те же права, что и у группы.

Еще один пример: запросим «длинный» вывод информации о файле с исходным программным кодом на языке программирования C:

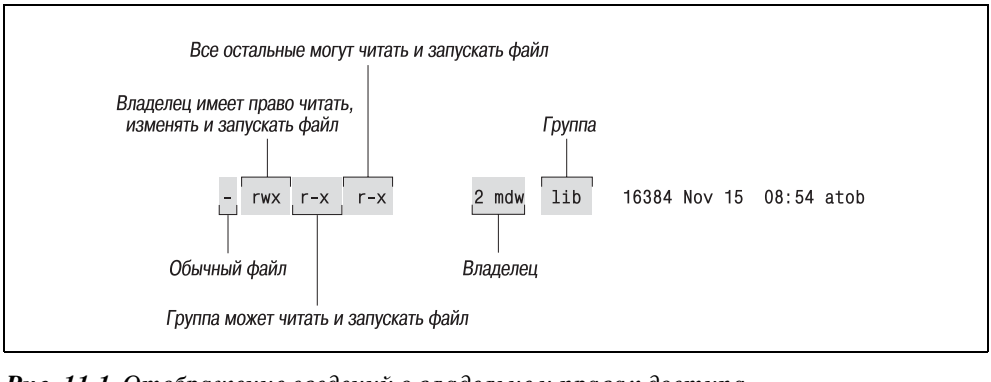


***Рис. 11.1.** Отображение сведений о владельце и правах доступа*

```
$ ls -l
-rw-rw-r--     1 kalle    kalle         12577 Apr 30 13:13 simc.c
```

Вывод показывает, что владелец, как и группа, обладает правами чтения и записи (`rw`). У всех остальных есть только право на чтение.

Предположим теперь, что мы скомпилируем файл и получим исполняемую программу. Компилятор *gcc* создаст файл *simc*:

```
$ gcc -osimc simc.c
$ ls -l
total 36
-rwxrwxr-x     1 kalle    kalle         19365 Apr 30 13:14 simc.c
-rw-rw-r--     1 kalle    kalle         12577 Apr 30 13:13 simc.c
```

В дополнение к битам разрешения чтения и записи *gcc* установил бит исполнения (`x`) для владельца исполняемого файла, группы и остальных. Это нужно сделать, чтобы файл можно было запустить:

```
$ ./simc
(вывод программы)
```

Еще пример – типичный каталог:

```
drwxr-xr-x     2 mdw      lib          512 Jul 17 18:23 perl
```

Здесь самый левый символ – `d`, что указывает на каталог. Разрешения на выполнение даны, поскольку вы хотите, чтобы все могли посмотреть, что лежит в каталоге.

Файлы иногда могут находиться в не вполне понятных состояниях, о которых здесь речь не идет; подробности можно прочесть на страницах справочного руководства, относящихся к команде *ls*. А сейчас пора посмотреть, как можно изменять владение и права доступа.

Изменение владельца, группы и прав доступа

Как было сказано ранее, чаще всего можно обойтись правами по умолчанию, устанавливаемыми системой. Но всегда есть исключения, особенно для системных администраторов. Возьмем простой пример. Предположим, вы создаете каталог для нового пользователя в */home*. Вы должны создавать его как суперпользователь, но, закончив создание, передать владение пользователю, иначе тот не сможет пользоваться файлами! (К счастью, есть возможность воспользоваться командой *adduser*, которая описывается выше, в разделе «Создание учетных записей»; она позаботится об изменении владельца файлов.)

Аналогично у некоторых утилит, таких как СУБД MySQL и News, есть собственные пользователи. Никто не входит в систему под именем *mysql* или *News*, но такие пользователи и группы должны существовать, чтобы утилиты могли работать, соблюдая необходимый уровень безопасности. Обычно последним шагом при установке программного обеспечения является изменение владельца, группы и прав доступа согласно указаниям в документации.

Команда *chown* изменяет владельца файла, а команда *chgrp* изменяет группу. В Linux только *root* может использовать *chown* для изменения владельца файла,

но любой пользователь может поменять группу на другую, к которой принадлежит сам.

Итак, после установки некой программы с именем *sampsoft* можно заменить как владельца, так и группу на *bin*, выполнив команды:

```
# chown bin sampsoft
# chgrp bin sampsoft
```

Можно сделать это за один шаг, используя точечную нотацию:

```
# chown bin.bin sampsoft
```

Синтаксис изменения прав доступа более сложен. Права доступа можно назвать «режимом» файла, и команда, которая его изменяет, называется *chmod*. Начнем изучение этой команды с простого примера. Скажем, вы написали на языке Perl или Tcl хорошую программу с именем *header* и хотите иметь возможность выполнить ее. Вы должны набрать следующую команду:

```
$ chmod +x header
```

Знак «плюс» означает «добавить разрешение», а `x` указывает, какое именно разрешение.

Если вы хотите снять разрешение на выполнение, используйте вместо плюса знак «минус»:

```
$ chmod -x header
```

Эта команда присваивает права доступа на всех уровнях – владельца, группы и остальных. Допустим, что вы тайно копите программы и не хотите, чтобы командой пользовался еще кто-либо, кроме вас. Нет, это слишком жестоко; скажем лучше: вы боитесь, что в сценарии слишком много ошибок, и хотите защитить других пользователей от неприятностей, пока сами не опробуете его. Можно присвоить право выполнения только себе самому командой

```
$ chmod u+x header
```

То, что находится перед плюсом, обозначает уровень права доступа, а то, что после него, – тип права доступа. Разрешение для пользователя (для себя) обозначается `u`, для группы – `g`, для остальных – `o`. Поэтому для присвоения права доступа себе самому и группе введите:

```
$ chmod ug+x header
```

Можно присваивать несколько прав доступа:

```
$ chmod ug+rwx header
```

Есть еще несколько сокращений, о которых вы можете узнать из страниц справочного руководства для *chmod*, чтобы поразить кого-нибудь, глядящего через плечо на вашу работу, но они не добавляют функциональности помимо той, которую мы вам показали.

Каким бы таинственным ни казался синтаксис аргумента режима, есть другой синтаксис, еще более сложный. Придется, однако, по некоторым причинам описать и его. Во-первых, есть ситуации, которые не охватываются только что продемонстрированным синтаксисом, называемым *символическим режимом*. Во-

вторых, в документации часто используется другой синтаксис, называемый *абсолютным режимом*. В-третьих, иногда абсолютный режим может оказаться более удобным.

Чтобы понять абсолютный режим, нужно думать на языке битов и восьмеричной нотации. Не пугайтесь, это не очень сложно. В обычном режиме есть три символа, относящиеся к уровням прав доступа (пользователь, группа и остальные). Эти уровни показаны на рис. 11.2. Внутри каждого уровня есть три бита, соответствующие правам на чтение, запись и исполнение.

Допустим, вы хотите дать себе право на чтение, а всем остальным не давать никаких прав. Для этого нужно задать бит, представляемый числом 400, поэтому команда будет такой:

```
$ chmod 400 header
```

Чтобы дать всем право на чтение, выберите нужный бит для каждого уровня: 400 для себя самого, 40 для группы и 4 для остальных. Полностью команда выглядит так:

```
$ chmod 444 header
```

Эта команда аналогична использованию режима *+r*, за исключением того, что одновременно снимаются все права записи и выполнения. (Если быть более точным, это все равно, что режим *=r*, о котором мы раньше не говорили. Знак равенства означает «присвоить эти права и никакие больше».)

Чтобы дать всем права на чтение и исполнение, нужно добавить биты чтения и исполнения. Например, 400 плюс 100 дает 500. Поэтому соответствующая команда будет следующей:

```
$ chmod 555 header
```

Это то же самое, что *=rx*. Для того чтобы дать кому-либо полный доступ, нужно задать соответствующую цифру равной 7 – сумме 4, 2 и 1.

И под конец еще один трюк: задание режима по умолчанию, присваиваемого каждому создаваемому вами (с помощью текстового редактора, оператора перенаправления `>` и тому подобного) файлу. Это делается командой `umask`, которую можно поместить и в ваш стартовый файл оболочки. Последний файл может называться *.bashrc*, *.cshrc* или как-то иначе в зависимости от используемой оболочки (мы обсуждали стартовые файлы в главе 4).

Рис. 11.2. *Биты в абсолютном режиме*

Использование аргумента команды `umask` схоже с абсолютным режимом `chmod`, но значение битов инвертировано. Нужно определить тип доступа, который вы хотите установить для владельца, группы и остальных, и вычесть каждую цифру из 7. В результате получится маска из трех цифр.

Например, вы хотите себе предоставить все права (7), своей группе – право чтения и выполнения (5), а остальным – никаких прав (0). Вычтите каждую цифру из 7 и получите 0 – для себя, 2 – для своей группы и 7 – для остальных. Поэтому в стартовый файл нужно поместить команду:

```
umask 027
```

Необычный прием, но работает. Команда `chmod` при интерпретации вашего режима смотрит на маску. Например, если при создании файла вы присваиваете ему исполняемый режим, право на исполнение будет присвоено вам, вашей группе и никому другому, поскольку маска не позволяет остальным иметь какой-либо доступ.

12

Установка, обновление и сборка программ

В этой главе мы покажем, как обновлять программное обеспечение. Хотя в большинстве дистрибутивов Linux есть автоматизированные средства установки, удаления и обновления отдельных программных пакетов системы, часто приходится устанавливать программное обеспечение вручную.

Не очень опытным пользователям проще всего устанавливать и обновлять программное обеспечение с помощью *системы пакетов*, предоставляемой большинством дистрибутивов. Если не пользоваться системой пакетов, то установка и обновление оказываются сложнее, чем в большинстве коммерческих систем. Даже если у вас есть скомпилированные двоичные файлы, возможно, потребуется распаковать их из архивного файла. Может также потребоваться создать символические ссылки или присвоить значения переменным окружения, чтобы двоичные файлы могли определить, где находятся необходимые им ресурсы. В других случаях требуется самостоятельная компиляция программного обеспечения из исходных текстов.

Обновление программного обеспечения

Linux находится в непрерывном развитии. Благодаря кооперативной природе проекта постоянно становится доступным новое программное обеспечение и обновляются версии программ.

Можно ли в такой ситуации рассчитывать, что у вас на машине всегда будут самые новые версии системного программного обеспечения? Если ответить коротко, то нельзя. В этом разделе мы обсудим, зачем и в какой момент следует обновлять систему, и покажем, каким образом можно обновить некоторые важные компоненты системы.

Когда нужно производить обновление? Обычно следует рассматривать вопрос об обновлении части системы, только если выявилась *необходимость* обновления. Например, если вы узнали о выпуске новой версии некоторого приложения, в которой исправлены важные ошибки (то есть ошибки, действительно влияющие на вашу работу с приложением), то вам, возможно, следует обновить это приложение. Если в новой версии есть новые функции, которые вы считаете полезными для себя, или она обеспечивает резкий рост производительности по сравнению

с текущей версией, тогда тоже есть смысл произвести обновление. Если ваша машина тем или иным способом подключена к Интернету, то причиной обновления может быть возможность заделать брешь в системе защиты, о которой вы недавно узнали. Однако делать обновление только для того, чтобы иметь последнюю версию какой-либо программы, не очень разумно.

Обновление иногда оказывается достаточно трудоемкой операцией. Например, вы можете пожелать обновить программу, для работы которой требуются новейшие версии компилятора, библиотек или другого программного обеспечения. Обновление этой программы может также потребовать обновления некоторых других компонентов системы, что займет много времени. С другой стороны, это может быть аргументом в пользу того, чтобы поддерживать на машине свежие версии программного обеспечения: если у вас уже стоят свежие версии компилятора и библиотек, обновить такую программу не составит труда.

Как узнать о выпуске новых версий программного обеспечения Linux? Лучше всего следить за объявлениями в телеконференции *comp.os.linux.announce* (см. раздел «Телеконференции Usenet» главы 1), где помещаются объявления о новых выпусках и другие важные сведения. При наличии доступа в Интернет можно загрузить программное обеспечение через FTP и установить его. Другим хорошим источником новостей о программном обеспечении Linux является сайт *http://www.freshmeat.net*.[1] Для большинства отдельных пакетов имеются списки рассылки, подписавшись на которые вы всегда будете оставаться в курсе выхода последних версий нужных вам пакетов.

При отсутствии доступа в Usenet или Интернет лучше всего следить за последними изменениями, приобретя подписку на CD-ROM. В результате раз в несколько месяцев вы будете получать на CD-ROM обновленную копию разных FTP-сайтов Linux. Такую услугу предоставляет ряд поставщиков Linux, и она полезна, даже если у вас есть доступ в Интернет.

Так мы переходим к следующему вопросу: какой метод обновления лучше? Некоторые считают, что, когда появляется новая версия их любимого дистрибутива, проще всего полностью обновить систему, установив все «с нуля». При этом не приходится беспокоиться о том, смогут ли различные версии программ работать совместно. При отсутствии доступа в Интернет это может быть действительно самым легким методом: если вы получаете новый CD-ROM только раз в два месяца, значительная часть вашего программного обеспечения может быть устаревшей.

Однако, по нашему мнению, переустановка отнюдь не является хорошим способом обновления. Большинство современных дистрибутивов Linux не предназначены для такого рода обновления, и полная переустановка может занять много времени. Кроме того, при таком обновлении вы обычно теряете все свои модификации и настройки системы, и вам придется делать резервные копии личных каталогов пользователей и других важных файлов, которые могут быть потеряны при

---

[1] Русскоязычные читатели могут воспользоваться, например, следующими ресурсами: *http://www.opennet.ru/prog/sml/*, *http://www.nixp.ru/soft/main*, *http://www.linux.org.ru/index.jsp*, *http://lafox.net/*, *http://linux.mensh.ru/products/dist-upgrade*, хотя ресурсов по Linux, в силу взрывоподобной динамики ее распространения за последние несколько лет, так много, что невозможно назвать какое-то подмножество «наилучших». – *Примеч. науч. ред.*

переустановке. Многие новички выбирают такой путь, поскольку он проще всего. На практике различия между версиями могут быть не столь велики, и в полной переустановке системы обычно нет необходимости. Ее можно избежать, если немного разбираться в том, как производить обновление.

Общая процедура обновления программного обеспечения

Как уже говорилось в предыдущем разделе, обычно легче и правильнее обновлять только те приложения, которые в этом действительно нуждаются. Например, если вы не используете редактор Emacs, зачем стремиться установить самую свежую его версию? Возможно, не стоит также стараться установить самые последние версии часто используемых приложений. Если какая-то программа работает нормально, нет особой необходимости ее обновлять.

Современные Linux-системы предоставляют различные способы обновления программного обеспечения, из которых одни являются ручными (самыми гибкими, но и самыми сложными), а другие – полностью автоматизированными. Мы рассмотрим ниже три различных технологии: систему пакетов RPM, систему пакетов Debian и ручной способ.

Хотим подчеркнуть, что пользоваться пакетами и системами пакетов действительно удобно, и даже если вы опытный пользователь, стоит применять эту технику, чтобы сберечь время для более интересных задач. Ниже приводится краткий список преимуществ пакетных систем:

- Все, что относится к программному пакету, находится в одном загружаемом файле.
- Программный пакет может быть удален целиком, не нарушая работоспособности других пакетов.
- Пакетные системы ведут учет зависимостей в базах данных, что позволяет автоматически отслеживать зависимости. Например, они могут сообщить о необходимости установки более новой версии некоторой библиотеки, требуемой для работы приложения, которое вы намерены установить (и отказаться удалить библиотечный пакет, если его библиотеки используются другими установленными пакетами).

Конечно, у систем пакетов есть и недостатки, часть которых мы отметим, когда будем рассказывать о системах пакетов RPM и Debian. Общая проблема в том, что, начав применять систему пакетов (что почти неизбежно, если использовать интерфейсы автоматизированной установки приложения), приходится уже все устанавливать с помощью пакетов, иначе невозможно следить за зависимостями. По той же причине не рекомендуется смешивать разные системы пакетов.

Каждый день обновляется какая-нибудь программа, которую вы, возможно, используете. К сожалению, очень часто это происходит потому, что в программе была обнаружена и исправлена серьезная уязвимость. Некоторые умудренные опытом системные администраторы настаивают на необходимости ежедневно проверять сообщения, связанные с безопасностью, и обновлять каждый пакет вручную, используя инструментальные средства, о которых рассказывается в этом разделе. Благодаря этому они в состоянии контролировать все аспекты системы

и могут гарантировать, что никакие изменения не нарушат работоспособность существующего программного обеспечения. Такой подход вполне оправдан на специализированных системах (таких, как серверы электронной почты или маршрутизаторы) и при ограниченном наборе программного обеспечения.

В случае систем универсального назначения такое постоянное обновление программ повседневного пользования быстро превратится в основное занятие. Поэтому все основные дистрибутивы предлагают возможность автоматизированного обновления системы. Некоторые из них будут рассмотрены в этой главе, а пока постараемся разобраться с общими принципами управления пакетами. Здесь будет рассказано о функционировании службы обновления, что становится очень важным, когда возникает потребность установить новое программное обеспечение или выполнить какие-то действия, не предусматриваемые этой службой.

Использование RPM

RPM (Red Hat Package Manager – менеджер пакетов Red Hat) является средством автоматизации установки двоичных программных файлов, которое запоминает список необходимых файлов, благодаря чему вы можете быть уверены в правильной работе программного обеспечения. Несмотря на свое название, RPM может использоваться не только с Red Hat, но и со многими другими дистрибутивами, включая SUSE. Использование RPM значительно облегчает установку и удаление программного обеспечения.

Главная идея RPM состоит в ведении базы данных пакетов и принадлежащих им файлов. При установке нового пакета информация о нем записывается в базу данных. Затем, когда вы пытаетесь удалить пакет, для каждого входящего в него файла RPM проверяет, нет ли других пакетов, использующих его. Если они есть, то рассматриваемый файл не удаляется.

Кроме того, RPM отслеживает зависимости. Каждый пакет может зависеть от одного или нескольких других пакетов. При установке пакета RPM проверяет, установлены ли уже пакеты, от которых зависит данный пакет. Если не установлены, то RPM сообщает об этом и отказывается устанавливать пакет.

Зависимости используются и при удалении пакетов: когда вы хотите удалить пакет, от которого зависят какие-либо другие пакеты, RPM тоже сообщает об этом и отказывается выполнить задание.

За удобства, предоставляемые RPM, приходится платить. Во-первых, разработчику существенно сложнее сделать пакет RPM, чем просто упаковать все файлы в *tar*-архив. Во-вторых, из пакета RPM нельзя извлечь один файл: вам приходится устанавливать все или ничего.

Если у вас уже стоит система RPM, то устанавливать RPM-пакеты очень легко. Допустим, у вас есть RPM-пакет с названием *SuperFrob-4.i386.rpm* (RPM-пакеты всегда имеют расширение *.rpm*; *i386* указывает на то, что это двоичный пакет, скомпилированный для компьютеров на базе микропроцессора Intel). Его можно установить командой:[1]

---

[1] Многие крупные пакеты (лучший тому пример – OpenOffice 2.1) состоят из нескольких (иногда многих) RPM-пакетов; в таких случаях их можно, естественно, устанавливать «все сразу»: `# rpm -i *.rpm`. – *Примеч. науч. ред.*

```
tigger # rpm -i SuperFrob-4.i386.rpm
```

Вместо `-i` можно использовать длинную версию названия этого параметра; выбирайте, что вам предпочтительнее:

```
tigger # rpm --install SuperFrob-4.i386.rpm
```

При нормальной установке на экран ничего не выводится. Если вы хотите, чтобы RPM был более словоохотлив, можете дать такую команду:

```
tigger # rpm -ivh SuperFrob-4.i386.rpm
```

В результате будет выведено имя пакета и ряд контрольных сообщений, позволяющих следить за ходом установки.

Если пакету, который вы пытаетесь установить, нужен другой пакет, который еще не установлен, вы получите сообщение примерно следующего содержания:

```
tigger # rpm -i SuperFrob-4.i386.rpm
failed dependencies:
        frobnik-2 is needed by SuperFrob-4
```

Увидев такое сообщение, вам придется начать поиски пакета *frobnik-2* и установить сначала его. Разумеется, и этот пакет в свою очередь может зависеть от других пакетов.

Если вы хотите обновить уже установленный пакет, воспользуйтесь параметром *-U* или *--update* (что представляет собой сочетание параметра *-i* с некоторыми другими параметрами):

```
tigger # rpm -U SuperFrob-5.i386.rpm
```

Удаление пакета производится с помощью параметра *-e* или *--erase*. При этом вы указываете не имя файла пакета (которого у вас уже может не быть), а имя пакета и номер его версии:

```
tigger # rpm -e SuperFrob-5
```

Кроме уже описанных параметров, посредством которых изменяется состояние вашей системы, есть параметр *-q*, позволяющий получить разнообразные данные обо всем, что записано в базе данных RPM и файлах пакетов. Вот несколько полезных применений параметра *-q*:

- Выяснить номер версии установленного пакета:

```
tigger# rpm -q SuperFrob
SuperFrob-5
```

- Получить список всех установленных пакетов:

```
tigger# rpm -qa
SuperFrob-5
OmniFrob-3
...
glibc-2.3.4-23.4
```

- Выяснить, какому пакету принадлежит файл:

```
tigger# rpm -qf /usr/bin/dothefrob
SuperFrob-5
```

```
tigger# rpm -qf /home/kalle/.xinitrc
file /home/kalle/.xinitrc is not owned by any package
(не принадлежит ни одному пакету)
```

- Вывести сведения о заданном пакете:

```
tigger# rpm -qi rpm
Name        : rpm              Relocations: (not relocatable)
Version     : 4.1.1                Vendor: SUSE LINUX Products GmbH,
Nuernberg, Germany
Release     : 208.2 Build Date: Sat 11 Jun 2005 01:53:04
AM CEST
Install date: Tue 28 Jun 2005 10:02:18 AM CEST Build Host: purcell.suse.de
Group       : System/Packages Source RPM: rpm-4.1.1-208.2.src.rpm
Size        : 5970541 License: GPL
Signature   : DSA/SHA1, Sat 11 Jun 2005 01:58:41 AM CEST, Key ID a84edae89c800aca
Packager    : http://www.suse.de/feedback
Summary     : The RPM Package Manager
Description :
RPM Package Manager is the main tool for managing the software packages
of the SuSE Linux distribution.
...
Distribution: SuSE Linux 9.3 (i586)
```

- Показать файлы, которые будут установлены для указанного пакета:

```
tigger# rpm -qpl SuperFrob-5.i386.rpm
/usr/bin/dothefrob
/usr/bin/frobhelper
/usr/doc/SuperFrob/Installation
/usr/doc/SuperFrob/README
/usr/man/man1/dothefrob.1
```

Мы показали только основные режимы работы, дополнением к которым служит большое число вспомогательных параметров. О них можно узнать на странице справочного руководства для команды *rpm*(8).

Если вы столкнулись с необходимостью установки RPM-пакета на такой системе, как Slackware или Debian, которые не основываются на RPM, ситуация несколько сложнее.

Вы можете либо использовать довольно очевидную в применении команду *alien*, способную преобразовывать пакеты из одного формата в другой и имеющуюся в большинстве дистрибутивов, либо самостоятельно создать базу данных RPM.

В последнем случае вам сначала нужно получить саму программу *rpm*. Ее можно загрузить с сайта *http://www.rpm.org*. Соберите ее и установите согласно имеющейся инструкции по установке. Если компилятор *gcc* у вас уже установлен, тогда никаких проблем не должно возникнуть. Следует отметить, что некоторые новейшие версии *rpm* испытывают определенные проблемы со стабильностью, таким образом, если в вашем дистрибутиве отсутствует *rpm*, тогда вам следует подойти к выбору версии с особой осторожностью, чтобы не столкнуться с неожиданными результатами. На данный момент версия 4.1.1 выглядит достаточно стабильной.

Следующей задачей является инициализация базы данных RPM. Дистрибутивы, содержащие RPM, проводят инициализацию автоматически, но в других системах вам придется выполнить команду

```
tigger # rpm --initdb
```

Эта команда создаст ряд файлов в каталоге */var/lib/rpm*. Каталог */var/lib* должен уже существовать, в противном случае создайте его командой *mkdir*.

Теперь можно устанавливать пакеты RPM обычным образом, но поскольку вы устанавливали базовые части системы, такие как библиотека C, без помощи RPM, то будете получать ошибки типа:

```
tigger # rpm -i SuperFrob-4.i386.rpm
failed dependencies:
        libm.so.5 is needed by SuperFrob-4
        libdl.so.1 is needed by SuperFrob-4
        libc.so.5 is needed by SuperFrob-4
```

так как эти файлы не внесены в базу данных RPM. Конечно, в действительности эти файлы есть в вашей системе, иначе большинство программ просто не смогли бы работать. Для того чтобы заставить RPM работать, необходимо сообщить программе, что она не должна обращать внимания на зависимости. Это можно сделать, задав параметр командной строки *--nodeps*:

```
tigger # rpm -i --nodeps SuperFrob-4.i386.rpm
```

Теперь RPM установит пакет без всяких жалоб. Конечно, он будет работать, только если установлены необходимые для него библиотеки. Один лишь факт использования ключа *--nodeps* не спасает, когда «зависимая» библиотека или программное обеспечение не установлены на машине.

Представленных сведений достаточно для управления вашей системой с помощью RPM. Если вы хотите узнать больше, почитайте страницу справочного руководства по команде *rpm* или обратитесь на сайт *http://www.rpm.org*.

Некоторые коммерческие компании предлагают платные службы по автоматизированному обновлению, основанные на RPM. Подписавшись на эти услуги, можно автоматически обновлять систему: такая служба определяет, какие новые пакеты появились, и устанавливает их. Если вы являетесь пользователем SUSE, то SUSE предоставит вам такую службу (под названием «YOU») бесплатно. Даже для дистрибутива Debian (его пакетная система описывается в следующем разделе) есть система автоматического обновления (описывается там же). Однако некоторые эксперты по безопасности считают, что такое автоматизированное обновление создает угрозу информационной безопасности.

Использование dpkg и apt

После *rpm* самым популярным менеджером пакетов для Linux является *dpkg*, который управляет архивами *.deb*. Как подсказывает имя, формат *.deb* ведет происхождение от дистрибутива Debian, но им также пользуются Ubuntu и Kubuntu, Libranet и Xandros. Подобно формату RPM формат *.deb* ведет учет зависимостей и файлов, что помогает обеспечить целостность системы.

Технические различия между двумя форматами весьма незначительны: несмотря на несовместимость форматов RPM и *.deb* (нельзя, например, непосредственно установить пакет Debian в систему Red Hat), можно воспользоваться утилитой *alien* и транслировать пакеты *.deb* для установки в других дистрибутивах (и обратно). Главное различие форматов в том, что пакеты *.deb* создаются с помощью инструментов, позволяющих обеспечить непротиворечивость их структуры и соответствие политике (главным образом Debian Policy Manual, которую можно найти в пакете *debian-policy*), что позволяет разработчикам создавать пакеты высокого качества.

Тогда как *dpkg* представляет собой интерфейс низкого уровня к менеджеру пакетов Debian, большинство функций обычно осуществляются через комплект программ *apt* либо интерфейсные программы, такие как *dselect*, *aptitude*, *gnome-apt*, *synaptic* и *KPackage*.

Установить пакет *.deb* в системе Debian очень просто. Например, если есть пакет с именем *superfrob_4-1_i386.deb*, его можно установить с помощью команды

```
tigger # dpkg -i superfrob_4-1_i386.deb
Selecting previously deselected package superfrob.
(Reading database ... 159540 files and directories currently installed.)
Unpacking superfrob (from superfrob_4-1_i386.deb) ...
Setting up superfrob (4-1) ...
```

Если для пакета *superfrob* отсутствует некоторый связанный с ним пакет, *dpkg* выводит предупредительное сообщение:

```
tigger # dpkg -i superfrob_4-1_i386.deb
Selecting previously deselected package superfrob.
(Reading database ... 159540 files and directories currently installed.)
Unpacking superfrob (from superfrob_4-1_i386.deb) ...
dpkg: dependency problems prevent configuration of superfrob:
 superfrob depends on frobnik (>> 2); however:
  Package frobnik is not installed.
dpkg: error processing superfrob (--install):
 dependency problems - leaving unconfigured
Errors were encountered while processing:
 superfrob
```

Этот вывод показывает, что для полной установки пакета потребуется *frobnik* версии 2 или более новый. (Файлы пакета установлены, но не будут работать, пока не будет также установлен пакет *frobnik*.)

В отличие от RPM, *dpkg* не делает различия между установкой нового пакета и обновлением существующего: в обоих случаях используется параметр *-i* (или *--install*). Например, чтобы обновить *superfrob* с помощью нового загруженного пакета *superfrob_5-1_i386.deb*, нужно просто ввести команду:

```
tigger # dpkg -i superfrob_5-1_i386.deb
(Reading database ... 159546 files and directories currently installed.)
Preparing to replace superfrob 4-1 (using superfrob_5-1_i386.deb) ...
Unpacking replacement superfrob ...
Setting up superfrob (5-1) ...
```

Чтобы удалить пакет, можно воспользоваться параметрами *-r* (*--remove*) или *-P* (*--purge*). Параметр *--remove* удаляет большую часть пакета, но сохраняет фай-

лы конфигурации, тогда как *--purge* удаляет также системные файлы с настройками. Например, полностью удалить *superfrob* можно командой:

```
tigger # dpkg -P superfrob
(Reading database ... 159547 files and directories currently installed.)
Removing superfrob ...
```

С помощью *dpkg* можно также выяснить, какие пакеты установлены в системе, если задать параметр *-l* (*--list*):

```
tigger $ dpkg -l
Desired=Unknown/Install/Remove/Purge/Hold
| Status=Not/Installed/Config-files/Unpacked/Failed-config/Half-installed
|/ Err?=(none)/Hold/Reinst-required/X=both-problems (Status,Err:uppercase=bad)
||/ Name           Version      Description
+++-= == == == == == == =-= == == == == == == =-= == == == == == == == == == == =
ii a2ps            4.13b-15     GNU a2ps 'Anything to PostScript' converter
ii aalib1          1.4p5-10     ascii art library
ii abcde           2.0.3-1      A Better CD Encoder
...
ii zlib1g-dev      1.1.3-19     compression library - development
```

Первые три строки объясняют значение первых трех колонок перед именем каждого пакета. Как правило, в них должно стоять *ii*, что соответствует корректной установке пакета. Если там что-то другое, следует выполнить команду *dpkg -audit*, которая объяснит, какая неполадка обнаружена в системе и как ее исправить.

Параметр *-l* можно использовать с именем пакета или шаблоном имени в стиле glob. Узнаем, например, какая версия *superfrob* установлена:

```
tigger $ dpkg -l superfrob
Desired=Unknown/Install/Remove/Purge/Hold
| Status=Not/Installed/Config-files/Unpacked/Failed-config/Half-installed
|/ Err?=(none)/Hold/Reinst-required/X=both-problems (Status,Err: uppercase=bad)
||/ Name           Version      Description
+++-= == == == == == == =-= == == == == == == =-= == == == == == == == == == == =
ii superfrob        4-1         The superfrobulator
```

С помощью *dpkg* можно узнать, какому пакету принадлежит конкретный файл:

```
tigger $ dpkg --search /bin/false
shellutils: /bin/false
tigger $ dpkg --search /home/kalle/.xinitrc
dpkg: /home/kalle/.xinitrc not found.
```

Можно также показать информацию об установленном пакете или архиве *.deb*:

```
tigger $ dpkg --status dpkg
Package: dpkg
Essential: yes
Status: install ok installed
Priority: required
Section: base
Installed-Size: 3156
Origin: debian
Maintainer: Dpkg Development <debian-dpkg@lists.debian.org>
Bugs: debbugs://bugs.debian.org
```

```
Version: 1.9.19
Replaces: dpkg-doc-ja
Pre-Depends: libc6 (>= 2.2.4-4), libncurses5 (>= 5.2.20020112a-1), libstdc++2.10-
glibc2.2 (>= 1:2.95.4-0.010810)
Conflicts: sysvinit (<< 2.80)
Conffiles:
 /etc/alternatives/README 69c4ba7f08363e998e0f2e244a04f881
 /etc/dpkg/dpkg.cfg 1db461ac9a1d4f4c8b47f5061078f5ee
 /etc/dpkg/dselect.cfg 190f7cf843556324495ef12759b752e3
 /etc/dpkg/origins/debian 24926c0576edec3e316fd9f6072b8118
Description: Package maintenance system for Debian
 This package contains the programs which handle the installation and
 removal of packages on your system.
 .
  The primary interface for the dpkg suite is the 'dselect' program;
 a more low-level and less user-friendly interface is available in
 the form of the 'dpkg' command.
  .
 In order to unpack and build Debian source packages you will need to
 install the developers' package 'dpkg-dev' as well as this one.
tigger $ dpkg --info reportbug_1.43_all.deb
 new debian package, version 2.0.
 size 66008 bytes: control archive= 1893 bytes.
      40 bytes,     2 lines       conffiles
    1000 bytes,    24 lines       control
     986 bytes,    15 lines       md5sums
    1014 bytes,    41 lines   *   postinst         #!/bin/sh
     147 bytes,     5 lines   *   postrm           #!/bin/sh
     416 bytes,    19 lines   *   prerm            #!/bin/sh
Package: reportbug
Version: 1.43
Section: utils
Priority: standard
Architecture: all
Depends: python
Recommends: python-newt
Suggests: postfix | mail-transport-agent, gnupg | pgp, python-ldap (>= 1.8-1)
Conflicts: python (>> 2.3), python-newt (= 0.50.17-7.1)
Installed-Size: 195
Maintainer: Chris Lawrence <lawrencc@debian.org>
Description: Reports bugs in the Debian distribution.
 reportbug is a tool designed to make the reporting of bugs in Debian
 and derived distributions relatively painless. Its features include:
 .
  * Integration with the mutt, af, and mh/nmh mail readers.
  * Access to outstanding bug reports to make it easier to identify
    whether problems have already been reported.
  * Support for following-up on outstanding reports.
  * Optional PGP/GnuPG integration.
 .
 reportbug is designed to be used on systems with an installed mail
 transport agent, like exim or sendmail; however, you can edit the
 configuration file and send reports using any available mail server.
```

С помощью *dpkg* можно также вывести список файлов и каталогов, включенных в архив *.deb*:

```
tigger $ dpkg --contents superfrob_4-1_i386.deb
-rwxr-xr-x root/root 44951 2002-02-10 12:16:48 ./usr/bin/dothefrob
-rwxr-xr-x root/root 10262 2002-02-10 12:16:48 ./usr/bin/frobhelper
...
```

dpkg, как и *rpm*, обладает многими другими параметрами. Подробности вы найдете на страницах справочного руководства *dpkg* и *dpkg-deb*.

Помимо *dpkg* дистрибутив Debian и другие, основанные на нем, предлагают комплект программ *apt*.[1] *apt* (сокращение от «advanced package tool») разрабатывался как независимая от способа архивирования система, способная работать с несколькими форматами пакетов. Одной из важнейших особенностей *apt* является его способность автоматически разрешать зависимости. Например, если *superfrob* требует вторую или более позднюю версию *frobnik*, *apt* постарается найти *frobnik* в доступных ему источниках (включая CD-ROM, локальные зеркала и Интернет).

Самым полезным интерфейсом к *apt* служит команда *apt-get*. Она управляет списком доступных пакетов («кэшем пакетов») и может быть использована для разрешения зависимостей и установки пакетов. Типичный сеанс работы с ней начинается с обновления кэша *apt*:

```
tigger # apt-get update
Get:1 http://http.us.debian.org stable/main Packages [808kB]
Get:2 http://http.us.debian.org stable/main Release [88B]
Hit http://non-us.debian.org stable/non-US/main Packages
Hit http://non-us.debian.org stable/non-US/main Release
Get:3 http://security.debian.org stable/updates/main Packages [62.1kB]
Get:4 http://security.debian.org stable/updates/main Release [93B]
Fetched 870kB in 23s (37kB/s)
Reading Package Lists... Done
Building Dependency Tree... Done
```

Этот вывод показывает, что имеются обновления для стабильной версии дистрибутива, поэтому можно обновить пакеты, которые уже установлены в системе. Чтобы осуществить такое обновление автоматически, можно воспользоваться параметром *upgrade*:

```
tigger # apt-get upgrade
The following packages have been kept back:
  gnumeric
17 packages upgraded, 0 newly installed, 0 to remove and 1 not upgraded.
Need to get 16.3MB of archives. After unpacking 5kB will be freed.
Do you want to continue? [Y/n] y
Get:1 http://http.us.debian.org stable/main base-passwd 3.4.6 [17.2kB]
Get:2 http://security.debian.org stable/updates/main ssh 1:3.1.6p4-1 [600kB]
...
(Reading database ... 159546 files and directories currently installed.)
Preparing to replace ssh 1:3.0.3p2-6 (using .../ssh_1%3a3.1.6p4-1_i386.deb) ...
```

[1] В некоторые дистрибутивы на базе RPM теперь включается *apt*, поскольку *apt* предназначен для работы с любым форматом пакетов.

```
Unpacking replacement ssh ...
...
```

Как можно заметить, в отличие от большинства команд Linux, для команд *apt* действия задаются без черточек. *apt-get* использует некоторые параметры, но они служат только для модификации поведения основной заданной операции.[1]

Обратите внимание, что пакет *gnumeric* не был обновлен автоматически, вероятно потому, что это потребовало бы установки дополнительных пакетов. Для того чтобы обновить его и разрешить зависимости, можно воспользоваться параметром *install* и указать имя одного или нескольких пакетов:[2]

```
tigger # apt-get install gnumeric
The following extra packages will be installed:
  libgal36 libglade3
The following NEW packages will be installed:
  libgal36
2 packages upgraded, 1 newly installed, 0 to remove and 0 not upgraded.
Need to get 8.3MB of archives. After unpacking 503kB will be used.
Do you want to continue? [Y/n] y
...
```

Еще одна удобная функция *apt* – нахождение информации о пакетах в хранилище. С помощью команды *apt-cache* осуществляется поиск сведений о доступных для установки пакетах. Часто с помощью *apt-cache* ищут пакеты по ключевым словам в описании пакета, используя слова, предложения (заключенные в кавычки) или регулярные выражения. Например, если нужно найти пакет, способный воспроизводить музыкальные файлы в формате Ogg Vorbis, можно воспользоваться параметром *search*:

```
tigger $ apt-cache search "ogg vorbis"
audacity - A fast, cross-platform audio editor
bitcollider-plugins - bitcollider plugins
cplay - A front-end for various audio players
gqmpeg - a GTK+ front end to mpg321/mpg123 and ogg123
libapache-mod-mp3 - turns Apache into a streaming audio server
libvorbis0 - The Vorbis General Audio Compression Codec
mp3blaster - Full-screen console mp3 and ogg vorbis player
mp3burn - burn audio CDs directly from MP3s or Ogg Vorbis files
oggtst - Read comments in ogg vorbis files
python-pyvorbis - A Python interface to the Ogg Vorbis library
vorbis-tools - Several Ogg Vorbis Tools
xmms - Versatile X audio player that looks like Winamp
xmms-dev - XMMS development static library and header files
mq3 - a mp3/ogg audio player written in Qt.
```

Если вас заинтересовал какой-либо из этих пакетов, можно узнать о нем поподробнее, воспользовавшись параметром *show* команды *apt-cache*:

---

[1] Есть и другие команды Linux, действующие так же, например *cvs*.

[2] Обратите внимание, что *apt-get* не устанавливает пакеты непосредственно из архивов *.deb*. Для установки архива *.deb* на компакт-диске или непосредственно загруженного из Интернета нужно использовать *dpkg* с параметром *--install*. При работе с *dpkg* все зависимости придется разрешать самостоятельно.

```
tigger $ apt-cache show xmms
Package: xmms
Priority: optional
Section: sound
Installed-Size: 4035
Maintainer: Josip Rodin <jrodin@jagor.srce.hr>
...
Description: Versatile X audio player that looks like Winamp
 XMMS (formerly known as X11Amp) is an X/GTK+ based audio player
 for various audio formats.
 .
 It's able to read and play:
  * Audio MPEG layer 1, 2, and 3 (with mpg123 plug-in),
  * WAV, RAW, AU (with internal wav plug-in and MikMod plug-in),
  * MOD, XM, S3M, and other module formats (with MikMod plug-in),
  * CD Audio (with CDAudio plug-in), with CDDB support,
  * .cin files, id Software,
  * Ogg Vorbis files.
 It has eSound, OSS, and disk writer support for outputting sound.
 .
 It looks almost the same as famous Winamp, and includes those neat
 features like general purpose, visualization and effect plug-ins,
 several of which come bundled, then spectrum analyzer, oscilloscope,
 skins support, and of course, a playlist window.
```

Рассмотрение всех функций *apt* выходит за рамки этой главы. Ответы на возникшие вопросы ищите на страницах справочного руководства *apt* (и тех, на которые она ссылается), а также в «APT HOWTO» (из пакета *apt-howto-en*).

В дополнение к утилитам командной строки разработан ряд удобных в работе текстовых и графических интерфейсов. К числу наиболее развитых из них принадлежит *KPackage*, входящий в состав графической среды KDE, но допускающий использование в других средах, например в GNOME. *KPackage* можно запустить из командной строки или через меню KDE – System (Система).

На рис. 12.1 показан внешний вид главного окна программы Kpackage, в левой части которого выводится список всех доступных для системы пакетов, а в правой части – информация о выбранном в списке пакете. Чтобы начать установку или удаление пакета, выберите пункт Install (Установить) или Uninstall (Удалить) в меню Packages (Пакеты) либо пометьте пакет щелчком на поле Mark (Выбор) и щелкните на кнопке Install marked (Установить выбранные) или Uninstall marked (Удалить выбранные). Пакеты *.deb* можно устанавливать непосредственно, если щелкнуть на кнопке Open (Открыть), расположенной на инструментальной панели в левой части экрана, и выбрать пакет для установки или перетащить значок *.deb* из окна файлового менеджера KDE в окно *KPackage*. В *KPackage* также есть средства для поиска пакетов с определенными именами. Как и во всех приложениях KDE, в *KPackage* можно получить подсказку, нажав клавишу F1 или выбрав пункт меню Help (Справка).

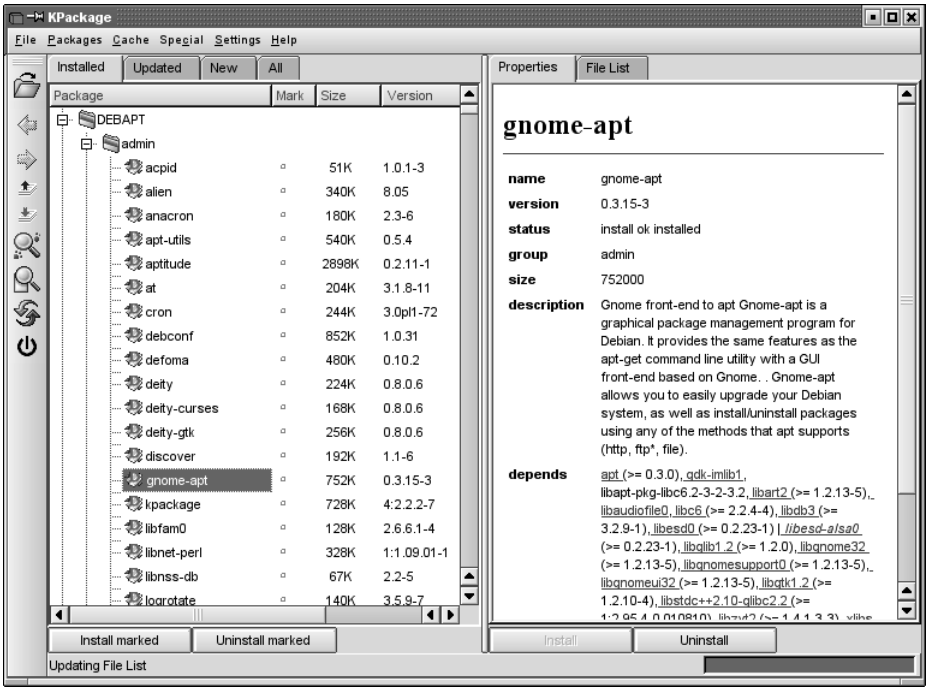


Рис. 12.1. *Менеджер пакетов KPackage*

Крупномасштабные и автоматизированные обновления

Почти все дистрибутивы включают в себя удобные механизмы обновления. В дистрибутиве SUSE один из таких механизмов поставляется как часть YaST. В Red Hat используется приложение с названием *up2date*, посредством которого осуществляется соединение с сетью обслуживания Red Hat Network. В дистрибутиве Debian имеется утилита *apt-get*, описанная в предыдущем разделе. Существуют и другие инструментальные средства, но, прежде чем ими можно будет воспользоваться, их надо будет установить, однако, чтобы решиться на это, обычно нет никаких причин.

Системы обновления разрабатывались таким образом, чтобы быть интуитивно понятными и простыми в использовании. Мы коротко расскажем о двух таких системах: YOU, распространяемой в составе дистрибутива SUSE, и ZENworks, которая пришла из мира Red Hat.

YaST Online Update: автоматизация обновления

Система YOU (YaST Online Update – система оперативного обновления YaST) – это автоматизированный инструмент обновления SUSE. Служба обновления является бесплатной (то есть ею может воспользоваться любой желающий без не-

обходимости оформлять платную подписку). Процедуру обновления можно запустить в любой момент по вашему желанию (хотя рекомендуется делать это регулярно, если вы вообще планируете использовать этот инструмент). Система обновления YOU интегрирована в набор инструментальных средств администрирования системы YaST; в разделе Software (Программное обеспечение) можно найти ярлык Online Update (Оперативное обновление). После щелчка на этом ярлыке будет открыт диалог обновления. В первый момент окно диалога остается пустым, поскольку ему требуется некоторое время, чтобы загрузить список доступных серверов. В течение длительного времени этот список может динамически изменяться. Затем из раскрывающегося списка Installation Source (Источник обновлений) можно выбрать ближайшее зеркало.

Если установить флажок Manually Select Patches (Отбирать обновления вручную) и щелкнуть на кнопке Next (Далее), то через некоторое время будет загружен список обновленных пакетов и будет открыта другая страница диалога (рис. 12.2), где можно будет отобрать пакеты для обновления. Те обновления, которые соответствуют текущему состоянию системы (другими словами, обновления для установленных в системе пакетов), уже будут помечены. Желательно все-таки просмотреть список до конца, потому что YOU может предложить обновления для некоторых пакетов, которые по юридическим причинам не были включены в состав установочных дисков. Например, можно будет загрузить и установить пакет *fetchmsttfonts*, содержащий шрифты TrueType от Microsoft (ну разве это не нелепо?). Другой пример – драйверы для различных карт WLAN, которые

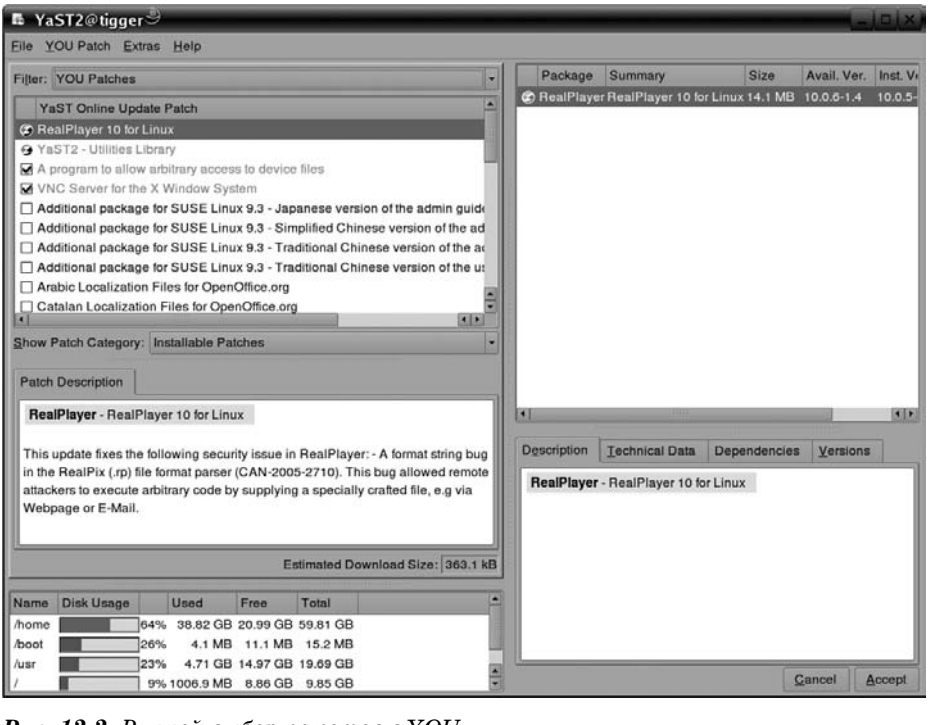


Рис. 12.2. *Ручной выбор пакетов в YOU*

Рис. 12.3. *Апплет SUSE Watcher демонстрирует наличие исправлений безопасности*

доступны только в виде обновлений. Так как фактически они не являются обновлениями существующих пакетов, по умолчанию они никогда не выбираются автоматически, таким образом, если возникнет необходимость установить подобные пакеты, их придется выбирать вручную, по крайней мере первый раз.

Если флажок Manually Select Patches (Отбирать обновления вручную) не был установлен, шаг выбора пакетов будет пропущен и сразу же начнется обновление.

В составе YOU поставляется один замечательный компонент – SUSE Watcher. Это апплет панели рабочего стола KDE, который занимается отслеживанием сайтов обновлений и выводит предупреждение (значок апплета приобретает вид красного шара), когда появляются новые исправления безопасности (рис. 12.3). Щелкнув правой кнопкой мыши на значке, можно вызвать контекстное меню апплета и с его помощью запустить процедуру обновления.

Red Carpet и ZENworks Linux Management: альтернативные системы управления пакетами

В этом разделе рассматривается еще одна система автоматизированного управления обновлениями, которая называется Red Carpet (ныне часть инструмента ZENworks Linux Management, разработанного компанией Novell), и описываются некоторые из преимуществ различных подходов к управлению пакетами.

Изначально разрабатывавшаяся как средство обновления программного обеспечения для рабочего стола GNOME, в настоящее время система Red Carpet распространяется компанией Novell в составе дистрибутива SUSE Linux, кроме того, она доступна из других источников и может устанавливаться как автономная система обновления. Она позволяет устанавливать обновления с различных серверов, включая репозитории *apt* и серверы ZENworks. Администраторы самых разных дистрибутивов Linux по достоинству оценят систему Red Carpet, которая реализует непротиворечивый интерфейс и набор команд, позволяющих универсальным образом управлять пакетами вне зависимости от конкретных особенностей отдельных дистрибутивов.

На стороне клиента система управления пакетами состоит из демона (*rcd*, в ближайшем будущем на смену ему придет *zmd*), интерфейса командной строки (*rug*) и графической оболочки. Все три части являются свободно распространяемым программным обеспечением, а вот серверная часть поставляется компанией Novell на коммерческой основе в составе полного пакета управления программным обеспечением. Серверная часть предназначена для крупных организаций и в данной книге рассматриваться не будет, но мы рассмотрим свободно распространяемую альтернативу инструментальных средств управления распространением программного обеспечения, которая называется Open Carpet.

Система ZENworks предназначена для распространения ПО Linux, разделяя его на различные каналы по аналогии с трансляцией телепередач. Каждый канал

представляет группу отдельных пакетов RPM, которые связаны между собой по некоторому признаку, например системное программное обеспечение или игры. Каждый пакет связывается с определенным разделом, как, например, Производительность или Мультимедиа, что позволяет быстро находить приложения, предназначенные для решения определенных задач. Существует возможность подписаться на определенные каналы практически так же, как в случае с кабельным телевидением, чтобы получать информацию только о том программном обеспечении, в котором вы заинтересованы. Это особенно удобно при наличии нескольких каналов, по которым распространяются различные версии одного и того же приложения, как, например, канал распространения стабильной версии Evolution и канал распространения нестабильных версий для разработчиков.

Система Red Carpet, как и прочие инструменты управления пакетами, способна разрешать все имеющиеся зависимости: так, например, если попытаться обновить Evolution до версии, которая требует более новой версии пакета gtkhtml, то система предложит обновить и этот пакет. Аналогичным образом, при попытке удалить программное обеспечение, которое требуется для обеспечения работоспособности других программ, Red Carpet предупредит об этом и при получении подтверждения удалит все пакеты, зависящие от выбранного. Например, если попытаться удалить пакет gtk+, система также удалит большую часть инструментальных средств рабочего стола GNOME, которые требуют наличия библиотеки gtk+.

Установка Red Carpet

Загрузить Red Carpet в виде пакета RPM можно с сайта *ftp://ftp.novell.com* или с веб-сайта производителя вашего дистрибутива Linux. Дополнительные ссылки можно попробовать поискать на сайтах *http://open-carpet.org* и *http://rpmfind.net*. Для работы системы потребуются пакеты *rcd* или *zmd* и, по крайней мере, либо *rug* (инструмент командной строки), либо графическая оболочка Red Carpet. Эти пакеты имеют некоторые зависимости, которые также придется установить командой *rpm -Uhv*.

Порядок работы с графической оболочкой Red Carpet

Графический интерфейс к системе обновления Red Carpet можно запустить либо с помощью главного меню рабочего стола, либо командой *red-carpet*. При первом запуске программа сообщит об отсутствии обновлений, поскольку еще ни на один канал не была оформлена подписка. Чтобы подписаться на какой-либо канал, нужно щелкнуть на кнопке Channels (Каналы) и выбрать требуемые элементы из предложенного списка.

После оформления подписки начальное окно Red Carpet будет отображать перечень новых версий пакетов, которые уже установлены в системе. Для каждого обновления отображаются название пакета, номер установленной версии и номер версии, доступной для загрузки, а также примечания о степени важности обновления, начиная с minor (незначительная) в случае незначительных изменений и заканчивая urgent (срочно) и necessary (обязательно) в случае исправления серьезных ошибок в системе безопасности.

Чтобы установить то или иное обновление, необходимо выделить требуемый пакет и щелкнуть на кнопке Mark for Installation (Пометить для установки) либо просто щелкнуть на кнопке Update All (Обновить все). Затем нужно щелкнуть на

кнопке Run Now (Запустить сейчас). После этого Red Carpet попросит подтвердить операцию и в случае положительного ответа исполнит ее.

В верхней части окна Red Carpet имеется ряд вкладок: Installed Software (Установленное программное обеспечение), Available Software (Доступное программное обеспечение) и Search (Поиск), которые позволяют просматривать более полные списки программного обеспечения. На первой вкладке выводится список программного обеспечения, которое уже установлено в системе, на второй – список еще не установленного, и на третьей – все программное обеспечение, которое известно системе. Все три списка могут быть отфильтрованы по каналам и разделам, а также по определенным словам в названиях пакетов и в описаниях.

Любой из пакетов может быть отмечен для установки или удаления на любой из первых четырех вкладок. Список действий, которые было решено выполнить, выводится в левой части окна, а также со всеми подробностями на вкладке Pending Actions (Запланированные действия). Операции по установке или удалению программного обеспечения будут выполнены только после щелчка на кнопке Run Now (Запустить сейчас) и подтверждения своих действий.

По окончании выполнения транзакций можно перейти на вкладку History (Хронология), чтобы ознакомиться с тем, как они исполнялись. Это поможет отыскать источник проблем, когда что-то пойдет не так после установки некоторого программного обеспечения, и откатить установку пакетов, вызвавших неполадки.

Если скорость загрузки или доступный перечень программного обеспечения не удовлетворяют вас, выберите пункт меню Edit (Правка)→Services (Службы) и добавьте или удалите те или иные серверы. Имеется возможность одновременного использования сразу нескольких служб, чтобы максимально разнообразить перечень доступного программного обеспечения. Список дополнительных служб можно найти на сайте *http://open-carpet.org*.

Порядок работы с командой rug

После того как вы разберетесь с основными принципами, заложенными в демон *zmd*, для вас будет более удобным обновление с помощью команды *rug*. Каждая операция будет содержать собственно команду *rug*, вслед за которой указываются название выполняемой операции и список дополнительных аргументов. Все операции имеют краткие и понятные названия. Мы не будем описывать все имеющиеся операции, подробнее с ними можно ознакомиться на страницах справочного руководства.

Обратите внимание: большинство операций, исполняемых с помощью команды *rug*, требуют наличия привилегий суперпользователя *root*, как и большинство других систем управления пакетами.

Первая операция, которая будет рассмотрена, – это получение списка доступных обновлений, *rug list-updates*. При желании установить их можно командой *rug update*. Чтобы отыскать некоторое программное обеспечение, необходимо ввести команду *rug search*, вслед за которой указать полное название искомого пакета или часть его. Для этих и всех остальных операций *rug* можно получить детальную справочную информацию, запустив команду *rug операция --help*. Полный список доступных операций можно получить с помощью команды *rug --help* или найти на страницах справочного руководства.

Из дополнительных операций, поддерживаемых командой *rug*, можно упомянуть операцию блокировки пакета, которая позволяет запретить возможность обновления того или иного пакета. Наложение блокировки выполняется с помощью команды *rug lock-add* *название-пакета*. Вывести перечень установленных блокировок с их порядковыми номерами можно командой *rug lock-list*, а снять блокировку можно командой *rug lock-delete* *номер-блокировки*.

Существуют также операции, с помощью которых можно выполнить проверку зависимостей пакетов. Например, команда *rug what-requires* *элемент* сообщит, какое ПО необходимо установить для обеспечения нормальной работоспособности указанного элемента. Зависимости можно проверять для всего чего угодно, включая целые пакеты, отдельные библиотеки, приложения, команды и т. д. Например:

```
system:/root # rug what-requires libusb
S | Channel      | Package           | Version          | libusb Version
--+--------------+-------------------+------------------+----------------
  | suse-92-i586 | ctapi-cyberjack   | 1.0.0-173.1      | (any)
i | suse-92-i586 | pcsc-cyberjack    | 1.1.1-245.1      | (any)
v | suse-92-i586 | pcsc-etoken       | 1.1.1-245.1      | (any)
```

В этом примере видно, какие пакеты используют библиотеку `libusb`. Первая колонка, `S`, описывает статус пакета. Символ `i` в строке с пакетом `pcsc-cyberjack` указывает, что данный пакет уже установлен. Отсутствие каких-либо символов говорит о том, что пакет не установлен, а символ `v` указывает, что установлена отличающаяся версия пакета. Вторая колонка содержит название канала, в рамках которого распространяется данный пакет, третья и четвертая колонки содержат название и номер версии пакета, и в последней колонке выводится номер версии библиотеки, которая требуется для пакета. В данном случае все три пакета требуют наличия любой версии библиотеки `libusb`.

Возможность проверки зависимостей на более низком уровне, чем уровень целых пакетов, обладает побочным эффектом, который позволяет устанавливать библиотеки командой *rug solvedeps*, не беспокоясь о версиях или пакетах. Например, если некоторое приложение требует наличия библиотеки `libfoo` версии выше 1.5, эту проблему можно решить командой *rug solvedeps "libfoo> 1.5"*. Существует возможность исключать учет зависимостей для пакетов, библиотек или исполняемых файлов с помощью символа `!`, добавляя его перед названием: *rug solvedeps "!libfoo" "frob> 2.3"*. Если существует возможность установить *frob* версии 2.3 или выше без установки библиотеки `libfoo`, то данная команда сделает это.

Наконец, с помощью команды *rug* можно получить доступ к многочисленным службам точно так же, как и в графической оболочке. Для этого следует вызвать команду *rug service-add*, в которой нужно указать URL службы. Некоторые из служб перечислены на сайте *http://open-carpet.org*.

Несколько пользователей

Представьте, что вы являетесь системным администратором и у вас возникает необходимость обновить сразу несколько систем, но при этом вы не обладаете привилегиями суперпользователя в этих системах. Как тогда поступить? Выход достаточно прост. В процессе установки системы, во время настройки, необходимо создать пользователя, который будет распознаваться демоном *zmd* как уда-

ленный пользователь. Тогда, даже если пароль пользователя *root* будет изменен, сохранится возможность устанавливать обновления.

Добавление пользователя можно выполнить с помощью команды *rug user-add имя-пользователя* или в графической оболочке выбрать пункт меню Edit (Правка) →Users (Пользователи). При этом потребуется указать пароль и наделить пользователя определенными привилегиями. Обратите внимание: эти имена пользователей и пароли не имеют никакого отношения к системе.

Пользователям можно выдать следующие привилегии:

Установка

Пользователь сможет устанавливать новое программное обеспечение.

Блокировка

Пользователь сможет добавлять и удалять блокировки.

Удаление

Пользователь сможет удалять программное обеспечение.

Подписка

Пользователь сможет изменять подписки на каналы.

Суперпользователь

Пользователь сможет получить полный доступ к системе обновления наравне с привилегиями, которыми обладает локальный пользователь *root*.

Доверительный

Пользователь сможет устанавливать неподписанные пакеты.

Обновление

Пользователь сможет обновлять установленное программное обеспечение.

Просмотр

Пользователь сможет просматривать списки установленного программного обеспечения и проверять наличие обновлений. Это единственная привилегия, которая включена по умолчанию.

После того как с помощью демона будет создана учетная запись, под ней можно будет управлять обновлением программного обеспечения без необходимости обладать полным доступом к системе.

Чтобы запретить возможность удаленного управления системой обновления, следует дать команду *rug set-prefs remote-enabled false*.

Чтобы предоставить возможность удаленного управления с использованием графической оболочки Red Carpet, следует выбрать пункт меню Edit (Правка)→Connect to Daemon (Соединение с демоном) и ввести адрес удаленного сервера. Чтобы то же самое сделать с помощью командной строки, нужно выполнить команду *rug*, которой следует передать адрес удаленного узла сети в параметре *--host*. Обратите внимание: по умолчанию демон принимает соединения на порте с номером 505.

Настройка локального сервера обновлений

В крупных компаниях нередко локальные сети защищены брандмауэрами и все обновления выполняются исключительно под контролем системного администт-

ратора. Для этого используются специализированные серверы обновлений уровня предприятия со сложными интерфейсами и многоступенчатыми уровнями привилегий администраторов. Мы не будем рассматривать в этой книге серверы такого рода. Если вы занимаетесь распространением небольшого числа обновлений всего для нескольких систем, или если вы занимаетесь разработкой некоторого программного обеспечения и стремитесь облегчить его установку и обновление, вам не нужно устанавливать сложную систему обновления. Единственное, что требуется в таких ситуациях, – это гарантировать доступ бета-тестеров к последней версии программы.

Open Carpet – это свободно распространяемый сервер, предназначенный для организации доступа к пакетам и метаданным по протоколам HTTP, FTP и Red Carpet. Это означает, что любой желающий сможет загрузить с вашего сервера файлы с помощью обычного веб-броузера и установить их вручную точно так же, как с самого обычного файлового сервера, но, кроме того, пользователи системы обновлений Red Carpet смогут обновлять программное обеспечение и разрешать зависимости полностью автоматически. В процессе работы вы обязательно столкнетесь с некоторыми шероховатостями, но, поиграв с файлами настройки, можно добиться вполне приличных результатов.

Чтобы создать свой сервер обновлений, необходимо установить пакеты *open-carpet* и *libredcarpet-python*, которые можно загрузить с сайта *http://open-carpet.org*, а также через официальную службу сайта Open Carpet. В эти пакеты входят примеры оформления файлов с настройками, которые обычно устанавливаются в каталог */usr/share/doc/packages/open-carpet/sample/*. В первую очередь следует отредактировать файл *server.conf*. В этом нет ничего сложного: нужно просто указать имя сервера, ваш адрес электронной почты и тому подобное. В заключение необходимо определить каталог канала. Для этого нужно создать каталоги, переписать в них требуемые пакеты и запустить команду *open-carpet*. Если все сложится нормально, вы получите работающий сервер обновлений. Чтобы отправить обновления пользователям, достаточно скопировать их в каталог канала и снова запустить сценарий.

Обновление другого программного обеспечения

Огромная часть программного обеспечения для Linux распространяется за рамками систем управления пакетами, хотя по достижении определенной популярности разработчики переходят на распространение своих продуктов в виде пакетов. Чтобы установить или обновить приложения, которые распространяются не в виде пакетов, необходимо загрузить их последние версии. Обычно их можно получить в виде *tar*-архива, сжатого с помощью *gzip* или *compress*. Такой пакет может быть представлен в нескольких формах. Чаще всего это *двоичные дистрибутивы*, в которых двоичные и другие связанные с ними файлы заархивированы и готовы к распаковке на машине. А также *дистрибутивы с исходными текстами*, когда предоставляется исходный программный код программного обеспечения или его части, и чтобы скомпилировать его и установить на машине, необходимо ввести некоторые команды.

Библиотеки совместного доступа облегчают распространение программного обеспечения в двоичной форме. Если версии установленных у вас библиотек совместимы с теми, что использовались при сборке программы, то все в порядке.

Однако во многих случаях проще (и правильнее) распространять программу в виде исходного программного кода. Это позволяет не только изучать программный код и развивать его дальше, но и собрать приложение специально для своей системы, причем с собственными библиотеками. Во многих программах можно задать определенные параметры компиляции, например включить в собираемую программу только необходимые вам функции. Такого рода настройка невозможна, если вы получаете готовые двоичные файлы.

При установке двоичных файлов без исходного кода возникают также проблемы безопасности. Хотя в UNIX-системах о вирусах почти не слышали, не столь сложно создать троянского коня[1] – программу, которая создает видимость полезных действий, а на самом деле наносит ущерб системе. Например, некто мог бы написать программу, имеющую такую «функцию», как уничтожение всех файлов в исходном каталоге пользователя, запустившего ее. Поскольку такая программа выполнялась бы с правами доступа запустившего ее пользователя, ничто не помешало бы ей произвести такое разрушительное действие. (Конечно, система безопасности UNIX не позволит повредить файлы других пользователей или какие-либо важные системные файлы, владельцем которых является *root*.)

Хотя наличие исходного кода не обязательно предотвращает такого рода ситуацию (не будете же вы читать исходный код каждой программы, которую компилируете на своей машине?), оно по крайней мере предоставляет возможность проверить, чем же в действительности занимается программа. Помимо того, если доступен исходный код, весьма вероятно, что найдутся люди, которые его изучат, поэтому пользоваться исходным кодом несколько безопаснее, но полностью полагаться на это нельзя.

Существуют определенные методики, позволяющие производить проверку двоичных пакетов, в частности цифровая подпись. Поставщик программного пакета может подписывать пакет своим ключом PGP, а инструментальные средства для работы с пакетами, такие как RPM, обладают возможностью проверять эти подписи. Однако в этом случае также придется полагаться на добросовестность поставщика программного пакета и отсутствие у него недобрых намерений. Единственное, что может гарантировать цифровая подпись, – это то, что пакет действительно получен от того, кто его подписал, и что по пути к жесткому диску пакет не претерпел никаких изменений.

В любом случае с дистрибутивами исходного программного кода или двоичных файлов работать очень легко. Если пакет имеет вид *tar*-архива, сначала определите вид архивации с помощью команды *tar t*. Если это двоичный дистрибутив, обычно его можно распаковать прямо в систему из корневого каталога / или из каталога */usr*.[2] При этом постарайтесь удалить прежние версии программы и со-

---

[1] В классическом случае «вирус» – это программа, прикрепляющаяся к «хозяину» и выполняемая при его запуске. В UNIX-системах, чтобы нанести какой-то ущерб, обычно нужны права суперпользователя, но если кто-то может завладеть его привилегиями, то ему, скорее всего, нет необходимости утруждать себя созданием вирусов.

[2] В последние годы расширилась практика устанавливать пакеты от сторонних производителей не в каталог */usr*, а в каталог */opt*, и многие находят именно это правильным. – *Примеч. науч. ред.*

провождающие ее файлы (которые не переписываются новым архивом). Если прежняя программа указана в пути поиска раньше, чем новая, у вас будет запускаться старая версия, пока вы ее не удалите.

С дистрибутивами исходного кода несколько сложнее. Прежде всего, нужно распаковать исходные файлы в отдельный каталог. В большинстве систем */usr/src* используется как раз для этой цели. Поскольку для сборки пакета, как правило, не требуются привилегии суперпользователя (такие права обычно нужны для установки программы после компиляции!), можно рекомендовать разрешить запись в */usr/src* всем пользователям, для чего надо выполнить команду:

```
chmod 1777 /usr/src
```

После этого любой пользователь сможет создавать подкаталоги в */usr/src* и помещать в них файлы. Первая цифра `1` в аргументе команды – это «липкий» (sticky) бит, предотвращающий удаление одним пользователем каталога, созданного другим.

Затем можно создать подкаталог в */usr/src* и распаковать в него tar-файл либо распаковать его прямо в */usr/src*, если в архиве указаны собственные подкаталоги.

Теперь, когда имеются исходные тексты, нужно прочесть все имеющиеся среди них файлы *README* и *INSTALL* или замечания по установке, содержащиеся в исходном программном коде. Такие документы почти всегда присутствуют в архивах. Основная методика сборки большинства программ такова:

1. Проверьте *Makefile*. В нем содержатся команды для утилиты *make*, которая управляет процессом создания программ компилятором. Во многих случаях требуется внести в *Makefile* мелкие изменения, чтобы настроить его на вашу систему; обычно они очевидны. В замечаниях по установке будет сказано, есть ли в этом необходимость. Если в пакете нет *Makefile*, то, возможно, вам придется его создать. Как это сделать, сказано в пункте 3.
2. Возможно, нужно отредактировать другие файлы, используемые программой. В некоторых приложениях требуется отредактировать файл с именем *config.h*. О том, как это сделать, тоже должно быть сказано в инструкции по установке.
3. Возможно, нужно запустить сценарий конфигурирования. Такой сценарий используют для определения возможностей машины, что необходимо для сборки более сложных приложений. В частности, если в каталоге верхнего уровня исходного программного кода нет файла *Makefile*, но вместо него есть файлы *Makefile.in* и *configure*, то пакет создан с помощью системы Autoconf. В таком случае (а это происходит все чаще) нужно выполнить сценарий конфигурирования:

   ```
   ./configure
   ```

   Точка и символ слэша (./) используются для того, чтобы запустить сценарий настройки *configure* из текущего каталога, а не из какого-нибудь другого, находящегося в переменной окружения `PATH`, в котором случайно может оказаться другая программа *configure*. В некоторых пакетах сценарию *configure* можно передавать дополнительные параметры, которыми подключаются или отключаются отдельные функции пакета (узнать, каковы эти параметры, можно с помощью команды *configure --help*). Когда сценарий *configure* отработает, можно переходить к следующему шагу.

4. Запустите *make*. Обычно при этом выполняются соответствующие команды компиляции, указанные в *Makefile*. Часто требуется указать *make* «цель», например *make all* или *make install*. Это часто встречающиеся цели. Первая обычно не нужна, но таким образом можно собрать все целевые программы, перечисленные в *Makefile* (например, если в пакет входит несколько программ, но по умолчанию компилируется только одна). Вторая часто используется для установки в системе исполняемых модулей и сопровождающих файлов после компиляции.[1] По этой причине *make install* обычно выполняется пользователем *root*.

   Между установкой программного обеспечения из исходных текстов и из двоичных пакетов существует одно существенное различие. При установке из пакетов с исходными текстами программы по умолчанию помещаются ниже каталога */usr/local*, который редко используется для установки двоичных пакетов.

При компиляции или установке в системе нового программного обеспечения могут возникнуть проблемы, особенно если эта программа не проверялась под Linux или зависит от других программ, которые у вас не установлены. В главе 21 мы подробно расскажем о компиляторе, *make* и сопутствующих инструментальных средствах.

Большинство пакетов помимо исходного кода и исполняемых модулей включают страницы справочного руководства и прочие файлы. Установочный сценарий (если таковой имеется) поместит эти файлы в нужное место. Что касается страниц справочного руководства, то вы обнаружите файлы с такими именами, как *foobar.1* или *foobar.man*. Обычно это исходные тексты в формате *nroff*, отформатированные таким образом, чтобы их можно было читать командой *man*. Если файл страниц руководства имеет числовое расширение, например *.1*, скопируйте его в каталог */usr/man/man1*, где *1* – это то число, которое используется в расширении имени файла. (Оно соответствует номеру раздела справочного руководства; чаще всего это 1.) Если файл имеет расширение типа *.man*, то обычно достаточно скопировать его в каталог */usr/man/man1*, изменив расширение *.man* на *.1*.

Обновление библиотек

Большинство программ в Linux компилируются в расчете на использование библиотек совместного доступа. В этих библиотеках содержатся полезные функции, использующиеся во многих программах. Вместо того чтобы хранить экземпляры этих функций во всех обращающихся к ним программах, их собирают в файлы системных библиотек, которые все программы считывают во время выполнения. Это означает, что при выполнении программы считывается код самой программы, а затем все подпрограммы из библиотечных файлов совместного доступа. Благодаря этому сберегается значительный объем дискового пространства: на диске хранится только один экземпляр библиотечной функции.

---

[1] Таким образом, чаще всего *make* необходимо выполнить последовательно два раза: сначала *make all* – для сборки пакета внутри выбранного каталога сборки, а затем *make install* – для установки созданных предыдущей командой компонент пакета в те места файловой системы, которые предполагал для них разработчик пакета. – *Примеч. науч. ред.*

Если вы используете систему управления пакетами, это означает, что вместе с программами автоматически устанавливаются все необходимые библиотеки. Система управления пакетами должна знать о зависимостях, связанных с библиотеками. Но поскольку различные программы могут зависеть от различных версий библиотек, а также потому, что вы могли установить программы в обход системы управления пакетами, иногда бывает необходимо понимать соглашения, принятые для библиотек, описываемые в этом разделе.

В некоторых случаях вместо использования разделяемых библиотек программа компилируется так, чтобы в ней был собственный экземпляр библиотечных функций (обычно это нужно при отладке). О собранных таким образом программах говорят, что в них использована статическая компоновка, в отличие от программ, использующих библиотеки совместного доступа, в которых применена динамическая компоновка.[1]

Таким образом, динамически компонуемые исполняемые модули требуют наличия на диске библиотек совместного доступа. Библиотеки совместного доступа реализуются так, чтобы компилируемые для их использования программы в целом не зависели от версий имеющихся библиотек. Это означает, что библиотеки можно обновлять, а все программы, собранные таким образом, чтобы использовать эти библиотеки, автоматически будут вызывать новые функции. (Существует исключение: если в библиотеках сделаны значительные изменения, старые программы не смогут с ними работать. Это видно, когда изменяется основной номер версии; более подробно мы расскажем об этом позже. В этом случае вам придется хранить и старые, и новые библиотеки. Все старые программы будут продолжать использовать старые библиотеки, а вновь скомпилированные программы будут использовать новые библиотеки.)

При сборке программы, использующей библиотеки совместного доступа, к ней добавляется участок кода, вызывающий при запуске программы динамический компоновщик *ld.so*. На *ld.so* возлагается задача найти требуемые для программы библиотеки совместного доступа и загрузить их функции в память. Кроме того, динамически компонуемые программы собираются с заглушками (stubs), которые просто занимают в выполняемых модулях место действительных функций из библиотек совместного доступа. Во время выполнения программы *ld.so* заменяет заглушки программным кодом, находящимся в библиотеках.

Для того чтобы получить список библиотек совместного доступа, от которых зависит данная исполняемая программа, можно воспользоваться командой *ldd*, например:

```
rutabaga$ ldd /usr/bin/X11/xterm
        linux-gate.so.1 => (0xffffe000)
        libXft.so.2 => /usr/X11R6/lib/libXft.so.2 (0x40037000)
        libfontconfig.so.1 => /usr/lib/libfontconfig.so.1 (0x4004b000)
        libfreetype.so.6 => /usr/lib/libfreetype.so.6 (0x40079000)
        libexpat.so.0 => /usr/lib/libexpat.so.0 (0x400e8000)
        libXrender.so.1 => /usr/X11R6/lib/libXrender.so.1 (0x40107000)
```

[1] Подробнее о создании и различиях в использовании библиотек динамической и статической компоновки можно почитать, например, здесь: *http://qnxclub.net/files/articles/libraries/libraries.html*. – *Примеч. науч. ред.*

```
libXaw.so.8 => /usr/X11R6/lib/libXaw.so.8 (0x4010f000)
libXmu.so.6 => /usr/X11R6/lib/libXmu.so.6 (0x4016b000)
libXt.so.6 => /usr/X11R6/lib/libXt.so.6 (0x40182000)
libSM.so.6 => /usr/X11R6/lib/libSM.so.6 (0x401d5000)
libICE.so.6 => /usr/X11R6/lib/libICE.so.6 (0x401dd000)
libXpm.so.4 => /usr/X11R6/lib/libXpm.so.4 (0x401f5000)
libXp.so.6 => /usr/X11R6/lib/libXp.so.6 (0x40205000)
libXext.so.6 => /usr/X11R6/lib/libXext.so.6 (0x4020d000)
libX11.so.6 => /usr/X11R6/lib/libX11.so.6 (0x4021c000)
libncurses.so.5 => /lib/libncurses.so.5 (0x40318000)
libutempter.so.0 => /usr/lib/libutempter.so.0 (0x4035d000)
libc.so.6 => /lib/tls/libc.so.6 (0x4035f000)
libdl.so.2 => /lib/libdl.so.2 (0x40478000)
/lib/ld-linux.so.2 (0x40000000)
```

Здесь мы видим, что программа *xterm* зависит от ряда библиотек совместного доступа, включая *libXaw*, *libXt*, *libX11* и *libc*. (Первые три, имена которых начинаются с *libX*, а также *libSM* и *libICE*, относятся к X Window System, причем *libc* является стандартной библиотекой языка C.) Видны также номера версий библиотек, для которых была скомпилирована программа (то есть версий использованных функций-заглушек), и имена файлов, в которых находятся библиотеки. Это те самые файлы, которые *ld.so* должен будет найти при выполнении программы. Первый файл в списке, с именем *linux-gate.so.1*, на самом деле не является библиотекой совместного использования, это так называемый динамический объект совместного использования, предоставляемый ядром. Его основное назначение – увеличить скорость обращения к системным вызовам в ядре и обеспечить приложения возможностью доступа к другим низкоуровневым функциям.

Для использования библиотеки совместного доступа версия заглушки (в исполняемом файле) должна быть совместима с версией библиотеки совместного доступа. Обычно библиотека является совместимой, если основные номера ее версии и версии заглушки совпадают. Основной номер версии – это та часть номера, которая находится сразу после *.so*. В данном случае *libX11* (самая главная библиотека, используемая X Window System) используется с основным номером версии 6. Файл библиотеки *libX11.so.6* (который обычно находится в */usr/X11R6/lib/*) вполне может быть просто символической ссылкой, например на *libX11.so.6.2*. Это означает, что основной номер версии библиотеки – 6, а дополнительный – 2. Поэтому, если программа компилировалась с версией заглушек 6.0, ее исполняемый модуль может использовать библиотеки версий 6.1, 6.2 и т. д. Если бы была выпущена новая версия библиотеки с основным номером версии 6 и дополнительным номером 3 (имя файла для нее было бы *libX11.so.6.3*), для ее использования достаточно было бы изменить символическую ссылку *libX11.so.6* так, чтобы она указывала на новую версию. В результате исполняемый модуль *xterm* мог бы автоматически воспользоваться исправлениями ошибок или другими преимуществами, присутствующими в новой версии. В разделе «Еще о библиотеках» главы 21 мы расскажем, как использовать библиотеки совместного доступа с вашими собственными программами.

В файле */etc/ld.so.conf* находится список каталогов, в которых *ld.so* ищет библиотечные файлы. Вот пример такого файла:

```
/usr/lib
```

```
/usr/local/lib
/usr/X11R6/lib
/opt/kde3/lib
```

Независимо от содержания *ld.so.conf* компоновщик *ld.so* всегда осуществляет поиск в */lib* и */usr/lib*.[1] Обычно не требуется изменять содержимое этого файла, а в переменной окружения `LD_LIBRARY_PATH` к этому пути поиска можно добавить другие каталоги (например, если у вас есть собственные библиотеки совместного доступа, которые не должны использоваться во всей системе). Однако, если вы все же добавите каталоги в */etc/ld.so.conf* либо обновите или добавите в систему дополнительные библиотеки, нужно будет выполнить команду *ldconfig*, которая заново создаст кэш библиотек совместного доступа в */etc/ld.so.cache* в соответствии с путем поиска для *ld.so*. Этот кэш используется *ld.so* для быстрого поиска библиотек во время выполнения вместо фактического просмотра указанных в пути каталогов. За дополнительными сведениями обратитесь к страницам справочного руководства по *ld.so* и *ldconfig*.

Теперь, когда вам стало ясно, как используются библиотеки совместного доступа, перейдем к задаче их обновления. Чаще всего обновляются две библиотеки: *libc* (стандартная библиотека языка C) и *libm* (библиотека математических функций). Поскольку имена этих библиотек несколько отличаются от общей схемы, мы рассмотрим здесь другую библиотеку – *libncurses*, которая «эмулирует» графическую оконную систему в текстовой консоли.

Каждой библиотеке совместного доступа соответствуют два отдельных файла:

<library>.a

Это статическая версия библиотеки. При статической компоновке программы функции копируются из этого файла непосредственно в выполняемый модуль, который в результате содержит собственный экземпляр библиотечных функций.[2]

<library>.so.<version>

Это образ самой библиотеки совместного доступа. При динамической компоновке в исполняемый модуль копируются функции-заглушки из этого файла, что позволяет *ld.so* во время выполнения программы найти библиотеку. При выполнении программы *ld.so* копирует функции из библиотеки совместного доступа в память, чтобы программа могла их использовать. Если программа компонуется динамически, файл *<library>.a* не используется.

Для библиотеки *libncurses* могут существовать файлы *libncurses.a* и *libncurses.so.5.4*. Файлы *.a* обычно находятся в */usr/lib*, а файлы *.so* хранятся в */lib*. При компиляции программы для ее сборки используется либо файл *.a*, либо файл *.so*, которые по умолчанию ищутся в */lib* и */usr/lib* (а возможно, и в других местах). Если у вас есть собственные библиотеки, вы можете хранить их где угодно,

---

[1] Но не ищет в текущем каталоге ./, что неочевидно и часто упускается из виду при разработке собственного программного обеспечения. – *Примеч. науч. ред.*

[2] В некоторых дистрибутивах статические версии библиотек помещаются в отдельный пакет и могут не устанавливаться по умолчанию. В таких случаях файлы *.a* вы обнаружите только после того, как установите их.

управляя их поиском при компоновке через параметр `-L`, передаваемый компилятору. (Подробнее в разделе «Еще о библиотеках» главы 21.)

Для большинства общесистемных библиотек образ *<library>.so.<version>* хранится в */lib*. Библиотеки совместного доступа могут находиться в любом из каталогов, в которых *ld.so* осуществляет поиск во время выполнения, в том числе */lib*, */usr/lib* и перечисленных в *ld.so.conf*. (Подробнее см. на странице справочного руководства по *ld.so*.)

Если вы посмотрите каталог */lib*, то обнаружите примерно такой список файлов:

```
lrwxrwxrwx   1 root   root     17   Jul 11 06:45 /lib/libncurses.so.5 \
    -> libncurses.so.5.4
-rwxr-xr-x   1 root   root   319472 Jul 11 06:45 /lib/libncurses.so.5.4
lrwxrwxrwx   1 root   root       13 Jul 11 06:45 libz.so.1 -> libz.so.1.2.2
-rwxr-xr-x   1 root   root    62606 Jul 11 06:45 libz.so.1.2.2
```

Здесь видны образы двух библиотек совместного доступа – *libncurses* и *libz*. Обратите внимание, что для каждого образа есть символическая ссылка в виде *<library>.so.<major>*, где *<major>* – основной номер версии библиотеки, а дополнительный номер опущен, поскольку *ld.so* ищет библиотеку только по основному номеру версии. Когда *ld.so* видит программу, скомпилированную с заглушками для *libncurses* версии 5.4, он ищет в своем пути поиска файл *libncurses.so.5*. В данном случае */lib/ libncurses.so.5* является символической ссылкой на */lib/ libncurses.so.5.4* – версию фактически установленной библиотеки.

При обновлении библиотеки необходимо заменить соответствующие ей файлы *.a* и *.so.<version>*. Заменить файл *.a* просто: скопируйте новую версию поверх него. Однако при замене образов библиотек совместного доступа *.so.<version>* нужно проявлять осторожность: многие программы текстового режима в системе зависят от этих образов, поэтому их нельзя так просто удалить или переименовать. Иными словами, символическая ссылка *<library>.so.<major>* должна всегда указывать на действительный образ библиотеки. Для этого сначала скопируйте новый файл образа в */lib*, а затем за один шаг измените командой `ln -sf` символическую ссылку так, чтобы она указывала на новый файл. Покажем это на примере.

Допустим, вы обновляете библиотеку *libncurses* с версии 5.4 на версию 5.5. У вас должны быть файлы *libncurses.a* и *libncurses.so.5.5*. Сначала скопируйте файл *.a* в соответствующее место, переписав прежнюю версию:

```
rutabaga# cp libncurses.a /usr/lib
```

Затем скопируйте новый файл образа в */lib* (или туда, где он должен находиться):

```
rutabaga# cp libncurses.so.5.5 /lib
```

Теперь после выполнения команды *ls -l /lib/libncurses* вы должны увидеть примерно следующее:

```
lrwxrwxrwx 1   root root       17 Dec 10 1999 /lib/libncurses.so.5 ->
libncurses.so.5.4
-rwxr-xr-x 1   root root   319472 May 11 2001 /lib/libncurses.so.5.4
-rwxr-xr-x 1   root root   321042 May 11 2001 /lib/libncurses.so.5.5
```

Измените символическую ссылку так, чтобы она указывала на новую библиотеку, командой

```
rutabaga# ln -sf /lib/libncurses.so.5.5 /lib/libncurses.so.5
```

В результате вы получите:

```
lrwxrwxrwx 1   root root       14   Oct 23 13:25 libncurses.so.5 ->\
/lib/libncurses.so.5.4
-rwxr-xr-x 1   root root   623620   Oct 23 13:24 libncurses.so.5.4
-rwxr-xr-x 1   root root   720310   Nov 16 11:02 libncurses.so.5.5
```

Теперь можно благополучно удалить старый файл образа – *libncurses.so.5.4*. Необходимо заменять символическую ссылку за один ход с помощью *ln -sf*, особенно при обновлении таких библиотек, как *libc*. Если бы вы сначала удалили ссылку, а затем попробовали создать ее снова командой *ln -s*, то, по всей вероятности, *ln* не смогла бы выполниться, поскольку символической ссылки уже нет и *ld.so* не смог бы найти библиотеку *libc*. Если ссылка уничтожена, то почти все программы в системе не смогут выполняться. При обновлении образов библиотек совместного доступа нужно проявлять большую осторожность. Для *libncurses* ситуация не так критична, потому что у вас всегда остаются программы командной строки, с помощью которых можно убрать учиненный беспорядок, но если вы привыкли пользоваться программами, основанными на *libncurses*, такими как Midnight Commander, то также могут возникнуть неудобства.

При обновлении библиотеки или добавлении библиотеки к системе очень неплохо запустить утилиту *ldconfig*, чтобы заново создать библиотечный кэш, используемый *ld.so*. В некоторых случаях *ld.so* не распознает новую библиотеку, пока не выполнена команда *ldconfig*.

Остается еще один вопрос: где брать новые версии библиотек? Некоторые основные системные библиотеки (*libc*, *libm* и т. д.) можно загрузить из каталога *ftp://ftp.gnu.org/pub/gnu/glibc*. Там содержатся исходные тексты библиотеки *libc* и некоторых других библиотек. Другие библиотеки поддерживаются и архивируются отдельно. В любом случае для всех устанавливаемых библиотек должны существовать файлы *.so.version*, а возможно, и файлы *.a* и комплект заголовочных файлов для использования с компилятором.

Обновление компилятора

Другой немаловажной частью системы, которую нужно поддерживать в обновленном состоянии, являются компилятор C и связанные с ним утилиты. В их число входят *gcc* (собственно компилятор GNU C и C++), компоновщик, ассемблер, препроцессор языка C и различные заголовочные файлы и библиотеки, используемые самим компилятором. Все они входят в состав дистрибутива *gcc* для Linux.[1] Обычно новая версия *gcc* выходит вместе с новыми версиями библиотеки *libc* и заголовочных файлов, причем все они тесно взаимосвязаны между собой.

---

[1] Но не во всех дистрибутивах компилятор *gcc* (и другие инструменты для разработки программ) устанавливается по умолчанию, особенно это участилось в последнее время с распространением «малых» конфигураций дистрибутивов, рассчитанных на начинающих или конечных пользователей. В таких случаях может потребоваться установить *gcc* дополнительно, например при помощи системы инсталляции пакетов, как описывалось выше. – *Примеч. науч. ред.*

Текущую версию *gcc* для Linux можно найти в разных архивах FTP, в том числе на сайте *ftp://ftp.gnu.org/pub/gnu/gcc*. В комментариях к выпуску должно быть сказано, как действовать. При отсутствии доступа в Интернет можно подписаться на получение CD-ROM с архивами FTP-сайта по обычной почте, как это было описано ранее.

Выяснить, какая версия *gcc* установлена, можно, выполнив команду

```
gcc -v
```

Она выдает примерно следующее:

```
Reading specs from /usr/lib/gcc-lib/i586-suse-linux/3.3.5/specs
Configured with: ../configure --enable-threads=posix --prefix=/usr --with-localprefix=/
usr/local --infodir=/usr/share/info --mandir=/usr/share/man --enablelanguages=
c,c++,f77,objc,java,ada --disable-checking --libdir=/usr/lib --enablelibgcj
--with-slibdir=/lib --with-system-zlib --enable-shared --enable-__cxa_atexit
i586-suse-linux
Thread model: posix
gcc version 3.3.5 20050117 (prerelease) (SUSE Linux)
```

Интересующая нас информация находится в последней строке – здесь указывается номер версии *gcc* и дата выпуска. Учтите, что *gcc* является лишь оболочкой к действительному компилятору и средствам генерации кода, находящимся в каталоге

```
/usr/lib/gcc-lib/machine/version
```

gcc (обычно находится в */usr/bin*) можно использовать с разными версиями собственно компилятора, используя ключ `-V`. В разделе «Программирование с использованием gcc» главы 21 мы подробно опишем порядок работы с компилятором *gcc*.

Здесь мы хотим предупредить вас, чтобы вы не пытались пользоваться более новыми компиляторами, если нет точного представления о том, для чего вам это нужно. Новые компиляторы могут генерировать объектные файлы, несовместимые со старыми, что может привести к серьезным неприятностям. На момент написания книги *gcc* версии 3.3.x считается стандартным компилятором для Linux, который должен подойти всем, хотя в свет уже вышли версии 3.4.0 и даже 4.0.0. Когда один из распространителей дистрибутивов (Red Hat) начал поставлять вместо него более новую версию (которая даже не была официальным выпуском), пользователи столкнулись с массой проблем. Конечно, когда вы будете читать эту книгу, стандартной, возможно, будет считаться другая версия компилятора. Если вы ищете приключений, попробуйте более новые версии, но будьте готовы к необходимости серьезных настроек системы.

Утилиты архивирования и сжатия

При установке или обновлении программного обеспечения на UNIX-системах прежде всего необходимо уметь работать со средствами сжатия и архивирования файлов. Имеются десятки утилит такого рода. Некоторые из них (такие как *tar* и *compress*) относятся по времени создания к самым ранним дням UNIX. Другие, такие как *gzip* и более новая *bzip2*, являются относительно новыми. Главной задачей этих утилит является архивирование файлов (то есть упаковка группы файлов в один для удобства перемещения и создания резервных копий) и сжатие

файлов для уменьшения дискового пространства, занимаемого файлом или группой файлов.

В данном разделе мы обсудим наиболее часто встречающиеся форматы файлов и утилиты, с которыми вы, вероятнее всего, столкнетесь. Например, в мире UNIX общепринято перемещать файлы или программное обеспечение в виде *tar*-архивов, сжатых с помощью *compress*, *gzip* или *bzip2*. Для того чтобы иметь возможность самостоятельно создавать или распаковывать такие файлы, нужно уметь пользоваться соответствующими инструментами. Эти инструменты чаще всего применяются при установке нового программного обеспечения или создании резервных копий и рассматриваются в двух последующих разделах данной главы. Пакеты, пришедшие из других миров, таких как Windows или Java, часто архивируются и сжимаются с помощью утилиты *zip*; такие архивы распаковываются с помощью команды *unzip*, которая имеется в большинстве инсталляций Linux.[1]

Использование gzip и bzip2

gzip является быстрой и эффективной программой, распространяемой в рамках проекта GNU. Основная функция *gzip* – сжать файл, сохранить сжатую версию в виде *filename.gz* и удалить исходный несжатый файл. Исходный файл удаляется только в случае успешного выполнения *gzip*. Случайное удаление файла при этом практически исключено. Конечно, *gzip*, являясь программным обеспечением GNU, имеет больше параметров, чем можно было бы ожидать, и характер работы этой программы можно во многом модифицировать с помощью параметров командной строки.

Предположим, что у нас есть большой файл с именем *garbage.txt*:

```
rutabaga$ ls -l garbage.txt
-rw-r--r--    1 mdw      hack     312996 Nov 17 21:44 garbage.txt
```

Для сжатия этого файла с помощью *gzip* мы просто выполним команду:

```
gzip garbage.txt
```

В результате *garbage.txt* будет заменен сжатым файлом *garbage.txt.gz*. Конечный результат следующий:

```
rutabaga$ gzip garbage.txt
rutabaga$ ls -l garbage.txt.gz
-rw-r--r--    1 mdw      hack      103441 Nov 17 21:44 garbage.txt.gz
```

Обратите внимание на то, что по завершении работы *gzip* файл *garbage.txt* удаляется.

Можно передать *gzip* список имен файлов. Каждый файл списка будет сжат и получит расширение *.gz*. (В отличие от программы *zip* для UNIX и MS-DOS, *gzip* по

---

[1] Обратите внимание: несмотря на похожесть названий *zip*, с одной стороны, и *gzip* и *bzip2* – с другой, эти утилиты имеют не так много общего. Утилита *zip* совмещает в себе функции упаковки и сжатия, тогда как утилиты *gzip*/*bzip2* используются исключительно для сжатия – обычно при работе с этими утилитами упаковка файлов производится с помощью утилиты *tar*. Форматы архивов, создаваемые этими утилитами, несовместимы, поэтому для распаковки определенных пакетов следует использовать соответствующую утилиту.

умолчанию не сжимает несколько файлов в один архив *.gz*. Для этой цели существует *tar*; подробнее об этом в следующем разделе.)

Эффективность сжатия файла зависит от его формата и содержания. Например, многие файлы графических форматов, такие как PNG и JPEG, уже в значительной мере сжаты, и применение *gzip* к таким файлам может не иметь эффекта. Обычно хорошо сжимаются текстовые файлы и такие двоичные файлы, как исполняемые модули и библиотеки. Сведения о сжатом файле можно получить при помощи команды *gzip -l*, например:

```
rutabaga$ gzip -l garbage.txt.gz
compressed  uncompr.  ratio uncompressed_name
   103115    312996   67.0% garbage.txt
```

Чтобы получить обратно оригинал из сжатой версии, мы используем *gunzip*, например:

```
gunzip garbage.txt.gz
```

После выполнения этой команды получаем:

```
rutabaga$ gunzip garbage.txt.gz
rutabaga$ ls -l garbage.txt
-rw-r-r--    1 mdw      hack     312996 Nov 17 21:44 garbage.txt
```

Полученный файл идентичен оригиналу. Обратите внимание, что после декомпрессии файла сжатая версия удаляется. Вместо *gunzip* можно воспользоваться командой *gzip -d* (например, если утилита *gunzip* не установлена).

gzip записывает имя исходного файла в его сжатую версию. Поэтому, если имя сжатого файла (вместе с расширением *.gz*) оказывается слишком длинным для выбранного типа файловой системы (например, если вы производите сжатие в файловой системе MS-DOS с именами формата 8.3), имя исходного файла может быть восстановлено *gunzip*, даже если при сжатии оно было усечено. Для декомпрессии файла с присвоением исходного имени используйте параметр *-N*. Чтобы убедиться в полезности этого параметра, рассмотрим последовательность команд:

```
rutabaga$ gzip garbage.txt
rutabaga$ mv garbage.txt.gz rubbish.txt.gz
```

Если теперь декомпрессировать *rubbish.txt.gz*, то в результате получим файл *rubbish.txt*, то есть с тем же именем, что и сжатый файл. Однако, задав параметр *-N*, мы получим:

```
rutabaga$ gunzip -N rubbish.txt.gz
rutabaga$ ls -l garbage.txt
-rw-r-r--    1 mdw      hack     312996 Nov 17 21:44 garbage.txt
```

gzip и *gunzip* могут также осуществлять компрессию и декомпрессию данных со стандартного ввода и вывода. Если не сообщить *gzip* имен файлов, подлежащих компрессии, то программа попытается сжимать данные со стандартного ввода. Аналогично *gunzip* с параметром *-c* выводит декомпрессированные данные на стандартное устройство вывода. Можно, например, передать по конвейеру выходные данные некоторой команды на *gzip*, чтобы за один шаг сжать выходной поток и записать его в файл:

```
rutabaga$ ls -laR $HOME | gzip > filelist.gz
```

Эта команда произведет вывод содержимого вашего исходного каталога с подкаталогами и запишет результат в сжатый файл *filelist.gz*. Содержимое файла можно вывести командой

```
rutabaga$ gunzip -c filelist.gz | more
```

Эта команда декомпрессирует файл *filelist.gz* и передаст результат по конвейеру команде *more*. Если вы используете *gunzip -c*, на диске остается сжатый файл.

Команда *zcat* аналогична *gunzip -c*. Можете рассматривать ее как версию *cat* для сжатых файлов. В Linux есть даже версия команды *less* для сжатых файлов, которая называется *zless*.

При сжатии файлов можно использовать параметры *-1*, *-2*, . . ., *-9*, чтобы задать скорость и степень сжатия. При этом *-1* (или *--fast*) задает самый быстрый метод, слабее сжимающий файлы, а *-9* (или *--best*) задает более медленный, но сильнее всего сжимающий файлы метод. По умолчанию значение метода сжатия равно *-6*. Эти параметры не влияют на использование команды *gunzip*, которая способна осуществить декомпрессию любого файла независимо от использованного метода сжатия.

В сравнении с более чем тридцатилетней историей развития операционной системы UNIX утилита *gzip* появилась относительно недавно. В большинстве UNIX-систем для сжатия файлов используются команды *compress* и *uncompress*, которые существовали в оригинальной Berkeley-версии UNIX. *compress* и *uncompress* очень похожи на *gzip* и *gunzip* соответственно. *compress* сохраняет сжатый файл как *filename.Z*, в отличие от *gzip*, сохраняющей сжатый файл как *filename.gz*, и использует несколько менее эффективный алгоритм сжатия.

Однако по ряду причин сообщество свободного программного обеспечения перешло на использование *gzip*. Во-первых, *gzip* работает лучше. Во-вторых, в результате спора по поводу авторских прав на алгоритм сжатия, используемый в *compress*, сторонним разработчикам может быть отказано в праве самостоятельной реализации алгоритма. Поэтому Фонд свободного программного обеспечения настаивал на переходе на *gzip*, что приветствовалось, по крайней мере, в сообществе Linux. *gzip* уже перенесена на некоторые архитектуры, за которыми последуют другие. Удачно, что *gunzip* может разархивировать файлы формата *.Z*, которые создает *compress*.

Появилась еще одна программа компрессии-декомпрессии – *bzip2*, перенявшая лидерство у *gzip*. *bzip2* – это новая программа, сжимающая в среднем на 10–20% лучше, чем *gzip*, за счет большей продолжительности процесса сжатия. Нельзя использовать *bunzip2* для декомпрессии файлов, сжатых *gzip*, и наоборот. Поскольку нельзя рассчитывать на то, что у всех установлена *bunzip2*, вам, возможно, следует ограничиться *gzip*, если вы собираетесь кому-либо посылать сжатые файлы. Однако стоит установить у себя *bzip2*, так как все большее число FTP-серверов начинает хранить пакеты, сжатые *bzip2*, с целью экономии дискового пространства и полосы пропускания. Весьма вероятно, что через несколько лет *gzip* станет такой же редкостью в мире UNIX, как утилита *compress* сейчас. Узнать файлы, сжатые *bzip2*, можно по обычному для них расширению *.bz2*.

Вообще говоря, параметры командной строки для *bzip2* и *gzip* отличаются друг от друга, но те, которые описаны в данном разделе, одинаковы. Более подробные сведения можно получить на странице справочного руководства для *bzip2*(1).

Подводя итоги, скажем, что для сжатия следует использовать *gzip*/*gunzip* или *bzip2*/*bunzip2*. Если вам встретится файл с расширением *.Z*, то он, скорее всего, был создан с помощью *compress*, и *gunzip* может выполнить его декомпрессию.

В старых версиях *gzip* для сжатых файлов использовалось расширение *.z* (нижний регистр), а не *.gz*. Из-за опасности путаницы с *.Z* оно было изменено. В любом случае *gunzip* поддерживает обратную совместимость с рядом расширений и типов файлов.

Использование tar

tar является универсальной утилитой архивирования, способной упаковать несколько файлов в один архивный файл, сохраняя при этом данные, необходимые для полного восстановления, такие как права доступа и владения. Название *tar* происходит от «tape archive», поскольку первоначально эта утилита предназначалась для архивирования файлов в виде резервных копий на магнитных лентах. Однако, как мы увидим, использование *tar* вовсе не ограничивается изготовлением резервных копий на магнитной ленте.

Команда *tar* имеет следующий формат:

```
tar functionoptions files...
```

Здесь *function* является буквой, указывающей выполняемую операцию, *options* является списком однобуквенных параметров этой функции, а *files* – списком упаковываемых или распаковываемых файлов в архиве. (Обратите внимание: символ *function* не отделяется пробелом от своих параметров.)

Параметр *function* может принимать следующие значения:

- `c` – создать новый архив;
- `x` – извлечь файлы из архива;
- `t` – перечислить содержание архива;
- `r` – дописать файлы в конец архива;
- `u` – заменить файлы в архиве более новыми;
- `d` – сравнить архивированные файлы с файлами файловой системы.

Чаще всего используются функции `c`, `x` и `t`.

Обычно используются следующие значения *options*:

k – сохранить при разархивации существующие файлы, то есть не заменять имеющиеся файлы извлекаемыми из архива;

f filename – задать имя архивного файла, который нужно прочесть или записать;

z – указать, что записываемые в архив файлы или имеющиеся в нем сжимаются с помощью *gzip*;

j – аналогично *z*, но вместо *gzip* используется *bzip2*. Доступна только в новых версиях *tar*. В некоторых промежуточных версиях для этих целей использовался символ *I*, более старые версии вообще не поддерживают возможность работы с *bzip2*;

v – заставить *tar* показать имена файлов, помещаемых в архив или извлекаемых из него. Полезно посмотреть с помощью этого значения, что в действительности происходит (если только вы не пишете сценарии оболочки, конечно).

Имеются и другие дополнительные параметры, о которых мы расскажем далее.

Синтаксис *tar* может вначале показаться сложным, но на практике он очень прост. Допустим, есть каталог *mt*, содержащий следующие файлы:

```
rutabaga$ ls -l mt
total 37
-rw-r--r--   1 root    root         24 Sep 21 2004 Makefile
-rw-r--r--   1 root    root        847 Sep 21 2004 README
-rwxr-xr-x   1 root    root       9220 Nov 16 19:03 mt
-rw-r--r--   1 root    root       2775 Aug 7 2004 mt.1
-rw-r--r--   1 root    root       6421 Aug 7 2004 mt.c
-rw-r--r--   1 root    root       3948 Nov 16 19:02 mt.o
-rw-r--r--   1 root    root      11204 Sep 5 2004 st_info.txt
```

Мы хотим упаковать содержимое этого каталога в один архивный *tar*-файл, для чего воспользуемся командой:

```
tar cf mt.tar mt
```

В данном случае первым аргументом *tar* является функция `c` (от «create»), за которой следуют параметры. Здесь мы используем один параметр `f mt.tar`, указывающий, что результирующий архивный файл должен иметь имя *mt.tar*. Последним аргументом является имя (или имена) архивируемого файла. В данном случае мы задаем имя каталога, чтобы *tar* упаковал в архив все файлы из этого каталога.

Обратите внимание, что первым аргументом должна быть буква функции, за которой следует список ее параметров. Поэтому нет смысла ставить дефис (`-`) перед параметрами, как того требуют многие команды UNIX. *tar* разрешает использовать дефис следующим образом:

```
tar -cf mt.tar mt
```

но в этом нет необходимости. В некоторых версиях *tar* первой буквой должна быть функция, то есть `c`, `t` или `x`. В других версиях порядок букв не является существенным.

Описанные здесь буквы функций следуют так называемому старому стилю параметров. Есть также более новый стиль коротких параметров, в котором параметрам функций предшествует дефис, и стиль длинных параметров, где используются длинные имена параметров с двумя дефисами перед ними. Более подробную информацию по этой теме вы найдете на страницах Info для *tar*.

Не забудьте указать имя файла, если используются буквы `cf`, иначе будет переписан первый файл в списке упаковываемых файлов, поскольку его имя по ошибке будет принято за имя архива!

Часто бывает удобно использовать *tar* с параметром `v`, что позволяет вывести перечень архивируемых файлов. Например:

```
rutabaga$ tar cvf mt.tar mt
mt/
mt/st_info.txt
mt/README
mt/mt.1
mt/Makefile
mt/mt.c
```

```
mt/mt.o
mt/mt
```

Если задать `v` несколько раз, выводятся дополнительные сведения, например:

```
rutabaga$ tar cvvf mt.tar mt
drwxr-xr-x root/root         0 Nov 16 19:03 2004 mt/
-rw-r--r-- root/root     11204 Sep 5 13:10 2004 mt/st_info.txt
-rw-r--r-- root/root       847 Sep 21 16:37 2004 mt/README
-rw-r--r-- root/root      2775 Aug 7 09:50 2004 mt/mt.1
-rw-r--r-- root/root        24 Sep 21 16:03 2004 mt/Makefile
-rw-r--r-- root/root      6421 Aug 7 09:50 2004 mt/mt.c
-rw-r--r-- root/root      3948 Nov 16 19:02 2004 mt/mt.o
-rwxr-xr-x root/root      9220 Nov 16 19:03 2004 mt/mt
```

Это очень удобно, так как позволяет убедиться в том, что действия *tar* соответствуют нашим ожиданиям.

В некоторых версиях *tar* буква `f` должна быть последней в списке параметров, поскольку за ней ожидается имя файла для чтения или записи. Если вообще не задать `f` *filename*, *tar* предполагает (в силу исторических причин), что нужно использовать устройство */dev/rmt0* (то есть первый ленточный привод). В разделе «Создание резервных копий» главы 27 мы поговорим об использовании *tar* с ленточными устройствами для создания резервных копий.

Теперь мы можем передать файл *mt.tar* другим пользователям для распаковки его на своих машинах. Сделать это они могут командой

```
tar xvf mt.tar
```

В результате выполнения этой команды создается подкаталог *mt*, в который помещаются все исходные файлы с теми же правами доступа, которые были в исходной системе. Владельцем новых файлов будет пользователь, выполнивший команду *tar xvf* (то есть вы), если только разархивацию не произвел *root*. В последнем случае сохраняется первоначальный владелец файла. Параметр `x` означает извлечение файлов, параметр `v` используется для перечисления всех извлекаемых файлов. Вывод будет таким:

```
courgette% tar xvf mt.tar
mt/
mt/st_info.txt
mt/README
mt/mt.1
mt/Makefile
mt/mt.c
mt/mt.o
mt/mt
```

Как можно видеть, *tar* сохраняет имя пути каждого файла относительно того расположения, где первоначально был создан архивный файл. Когда мы создавали архив командой *tar cf mt.tar mt*, единственным заданным именем входного файла было *mt* – имя содержащего файлы каталога. Поэтому *tar* записывает в архивный файл сам каталог и все находящиеся в нем файлы. Когда мы извлекаем файлы из архива, создается каталог *mt*, в который помещаются все файлы, что является процедурой, прямо противоположной созданию архива.

По умолчанию *tar* извлекает из архива все файлы относительно текущего каталога, в котором вы запускаете *tar*. Например, если бы вы собрались упаковать содержимое каталога */bin* командой:

```
tar cvf bin.tar /bin
```

то получили бы предупреждение:

```
tar: Removing leading / from absolute pathnames in the archive.
     (удаление ведущих / из абсолютных путей в архиве)
```

Это означает, что файлы будут сохранены в архиве в подкаталоге *bin*. При распаковке каталог *bin* создается в рабочем каталоге *tar*, а не как */bin* в системе, в которой производится распаковка. Это очень важно и имеет целью предотвращение катастрофических ошибок при распаковке. В противном случае распаковка файла, содержащего упакованный */bin*, привела бы к уничтожению вашего текущего каталога */bin*.[1] Если вам действительно нужно распаковать такой архив в */bin*, то это нужно делать, находясь в корневом каталоге /. Можно переопределить такое поведение при создании *tar*-архива с помощью параметра `P`, но делать так не рекомендуется.

Другим способом создания архива *mt.tar* был бы переход командой *cd* непосредственно в каталог *mt* и использование такой команды, как

```
tar cvf mt.tar *
```

При таком способе подкаталог *mt* не записывается в архивный файл, и при распаковке файлы помещаются в текущий рабочий каталог. Одна из тонкостей этикета работы с *tar* состоит в том, что всегда нужно упаковывать файлы при помощи *tar* таким образом, чтобы они содержали подкаталог, как мы делали это в первом примере с *tar cvf mt.tar mt*. Благодаря этому при распаковке архива создается также и каталог, в который помещаются файлы. Это позволяет избежать путаницы, которая могла бы возникнуть при помещении распакованных файлов непосредственно в рабочий каталог. Кроме того, пользователь освобождается от необходимости создания отдельного каталога для распаковываемых файлов. Конечно, есть масса ситуаций, когда вам это не потребуется. Но этикет таков, каков он есть.

При создании архивов можно указать утилите *tar* список подлежащих упаковке файлов и каталогов. В первом примере мы передали *tar* единственный каталог *mt*, но затем мы воспользовались символом `*`, который оболочка превращает в список всех имен в текущем каталоге.

Прежде чем распаковывать *tar*-архив, неплохо посмотреть на таблицу его содержимого и определить, как он был упакован. В результате вы можете решить, нужно ли создавать подкаталог для распаковки архива. Команда

```
tar tvf tarfile
```

выводит таблицу содержимого архива с именем *tarfile*. Обратите внимание: при использовании функции `t` требуется только один параметр `v`, чтобы получить «длинную» распечатку перечня файлов, как показано в следующем примере:

```
courgette% tar tvf mt.tar
```

[1] Некоторые старые реализации UNIX (например, Sinix и Solaris) именно так и поступали.

```
drwxr-xr-x root/root        0 Nov 16 19:03 2004 mt/
-rw-r--r-- root/root    11204 Sep 5 13:10 2004 mt/st_info.txt
-rw-r--r-- root/root      847 Sep 21 16:37 2004 mt/README
-rw-r--r-- root/root     2775 Aug 7 09:50 2004 mt/mt.1
-rw-r--r-- root/root       24 Sep 21 16:03 2004 mt/Makefile
-rw-r--r-- root/root     6421 Aug 7 09:50 2004 mt/mt.c
-rw-r--r-- root/root     3948 Nov 16 19:02 2004 mt/mt.o
-rwxr-xr-x root/root     9220 Nov 16 19:03 2004 mt/mt
```

Распаковка при этом не производится, мы видим только таблицу содержимого архива. Имена файлов говорят о том, что файл архива был создан из всех файлов подкаталога *mt*, и при распаковке архива будет создан подкаталог *mt*, в который будут помещены все файлы.

Можно извлекать из архива и отдельные файлы. Для этого служит команда

```
tar xvf tarfile files
```

где *files* является списком извлекаемых из архива файлов. Если не указывать имен, *tar* распаковывает весь архив.

При указании отдельных файлов, которые нужно извлечь, следует задавать полный путь в таком виде, как он указан в архиве. Например, если потребовалось извлечь из предыдущего архива только файл *mt.c*, это можно сделать командой:

```
tar xvf mt.tar mt/mt.c
```

В результате создается подкаталог *mt*, в который помещается файл *mt.c*.

Утилита *tar* имеет много параметров помимо перечисленных здесь. Мы говорили лишь о функциях, которыми вы будете пользоваться чаще всего, однако GNU-версия *tar* (особенно она) имеет функции, делающие ее идеальной для создания резервных копий и тому подобных задач. Более подробные сведения можно получить из страниц справочного руководства по *tar* и следующего раздела.

Использование tar совместно с gzip и bzip2

tar не осуществляет компрессии данных, сохраняемых в его архивах. Если вы создаете *tar*-архив из трех файлов по 200 Кбайт, то получите архив размером около 600 Кбайт. Очень часто *tar*-архивы сжимаются с помощью *gzip* (или более старых программ компрессии). Сжатый *tar*-архив можно создать, выполнив команды:

```
tar cvf tarfile files...
gzip -9 tarfile
```

Однако эта процедура весьма утомительна и требует выделить достаточный объем памяти для хранения несжатого архивного файла, прежде чем подвергнуть его обработке *gzip*.

Гораздо более ловкий способ выполнить ту же задачу – использовать функцию *tar*, позволяющую вывести архив на стандартное устройство вывода. Если задать архивный файл для чтения или записи как -, то данные будут считываться со стандартного устройства ввода и выводиться на стандартное устройство вывода. Например, можно создать сжатый архивный файл с помощью команды:

```
tar cvf - files... | gzip -9 > tarfile.tar.gz
```

В этом примере *tar* создает архив из файлов с перечисленными именами и выводит его на стандартное устройство вывода; затем *gzip* читает данные из стандартного устройства ввода, сжимает их и выводит на свое стандартное устройство вывода; наконец, мы перенаправляем сжатый архивный файл в *tarfile.tar.gz*.

Распаковать такой архивный файл можно с помощью команды:

```
gunzip -c tarfile.tar.gz | tar xvf -
```

gunzip декомпрессирует указанный архивный файл, выводит результат на стандартное устройство вывода, откуда он считывается утилитой *tar*, как со стандартного ввода, и распаковывается. Ну разве UNIX не прелесть?

Конечно, обе эти команды печатать долго. К счастью, GNU-версия *tar* имеет параметр `z`, задание которого автоматически создает или распаковывает сжатые архивы. (Мы приберегли разговор об этом параметре до настоящего момента, чтобы вы могли по достоинству оценить его удобство.) Например, для создания сжатых архивных файлов и их распаковки можно использовать команды:

```
tar cvzf tarfile.tar.gz files...
```

и

```
tar xvzf tarfile.tar.gz
```

Обратите внимание, что создаваемым таким образом файлам нужно присваивать расширение *.tar.gz* (или столь же часто используемое *.tgz*, которое удобно использовать в системах с ограничениями на имена файлов), чтобы был ясен их формат. Параметр `z` нормально работает и с другими функциями *tar*, такими как `t`.

Поддержка параметра `z` есть только в GNU-версии *tar*. При использовании *tar* в других UNIX-системах для выполнения таких же задач вам может потребоваться использование более длинных команд. Почти все системы Linux используют GNU-версию *tar*.

Если вы хотите использовать *tar* вместе с *bzip2*, то о своем выборе программы сжатия вы можете сообщить следующим образом:

```
tar cvf tarfile.tar.bz2 -use-compress-program=bzip2 files...
```

или короче:

```
tar cvf tarfile.tar.bz2 --use=bzip2 files...
```

или еще короче:

```
tar cvjf tarfile.tar.bz2 files
```

Последняя версия работает только с последними версиями GNU *tar*, поддерживающими параметр `j`.

Имея это в виду, можно писать короткие сценарии оболочки или псевдонимы для создания архивных файлов и их распаковки. Если вы пользуетесь *bash*, то можно включить в файл *.bashrc* следующие функции:

```
tarc () { tar czvf $1.tar.gz $1 }
tarx () { tar xzvf $1 }
tart () { tar tzvf $1 }
```

Если определены такие функции, то создать сжатый архивный файл из отдельного каталога можно с использованием команды

```
tarc directory
```

Полученный архивный файл будет назван *directory.tar.gz*. (Следите, чтобы в конце имени каталога не было символа слэша, иначе архив будет создан как *.tar.gz* в данном каталоге.) Чтобы вывести список файлов сжатого *tar*-архива, выполните команду:

```
tart file.tar.gz
```

Для распаковки такого архива выполните команду

```
tarx file.tar.gz
```

В заключение отметим, что файлы, созданные с помощью *gzip* и (или) *tar*, могут распаковываться хорошо известной утилитой *WinZip* в Windows. *WinZip* пока не поддерживает *bzip2*. И наоборот, файл в формате *.zip* можно распаковать на машине под Linux с помощью команды *unzip*.

Приемы работы с tar

Поскольку *tar* записывает в архив права владения и доступа к файлам, а также сохраняет полную структуру каталогов, символические и жесткие ссылки, с его помощью очень удобно копировать или перемещать целое дерево каталогов из одного места в другое в пределах данной системы и даже, как мы увидим позднее, с одной машины на другую. Используя описанный выше синтаксис с минусом, можно вывести *tar*-архив на стандартное устройство вывода, откуда он читается и распаковывается в любое место через стандартное устройство ввода.

Допустим, имеется каталог с двумя подкаталогами – *from-stuff* и *to-stuff*. Каталог *from-stuff* содержит целое дерево файлов, символических ссылок других элементов, которые трудно точно отобразить, используя команду *cp* с рекурсией. Чтобы скопировать все дерево из каталога *from-stuff* в каталог *to-stuff*, можно выполнить такие команды:

```
cd from-stuff
tar cf - . | (cd ../to-stuff; tar xvf -)
```

Просто и элегантно, не правда ли? Мы начинаем работать в каталоге *from-stuff* и создаем архив текущего каталога, который выводится на стандартное устройство вывода. Этот архив читается подоболочкой (содержащиеся в скобках команды); подоболочка совершает переход в целевой каталог *../to-stuff* (относительно *from-stuff*) и выполняет *tar xvf*, осуществляя чтение со стандартного устройства ввода. Никаких архивных файлов на диск не пишется; данные посылаются через канал от одного процесса *tar* другому. Второй процесс *tar* запускается с параметром `v`, благодаря чему выводятся имена всех распаковываемых файлов, что позволяет контролировать правильность работы команды.

С помощью такого приема можно перемещать целые деревья каталогов с одной машины на другую по сети. Для этого нужно лишь включить соответствующую команду *rsh* (или *ssh*) в подоболочку с правой стороны канала. Удаленная оболочка запустит *tar*, чтобы считать архив со стандартного устройства ввода. (Фактически GNU *tar* имеет возможность автоматически читать или выводить *tar*-архивы на других машинах через сеть; подробности вы найдете на страницах справочного руководства *tar*(1).)

13

Подключение к сети

Итак, вы застолбили свой участок во владениях Linux – установили и сконфигурировали систему. Что дальше? В какой-то момент вы захотите связаться с другими системами – Linux и прочими, и голубиная почта для этого не очень подходит.

К счастью, Linux поддерживает множество методов обмена данными и работы в сети. Главным образом это означает взаимодействие с другими системами в сетях TCP/IP, но существует также возможность организовать обмен информацией через последовательные линии связи и даже через радиоэфир. В этой главе мы рассмотрим способы настройки системы для связи с внешним миром.

Полным руководством по настройке Linux для работы в сетях по различным протоколам, в том числе и TCP/IP, является «Linux Network Administrator's Guide»[1], входящее в Проект документирования Linux и опубликованное издательством O'Reilly.

Сетевые взаимодействия по протоколу TCP/IP

Linux обеспечивает полную реализацию TCP/IP (Transmission Control Protocol/Internet Protocol – сетевой протокол управления передачей/межсетевой протокол). TCP/IP оказался наиболее удачным механизмом работы в компьютерных сетях по всему миру. Располагая Linux и сетевой картой, можно подключить свой компьютер к локальной сети (LAN) или к Интернету – Всемирной сети, работающей на TCP/IP (при соответствующем сетевом подключении).

Построить небольшую локальную сеть из UNIX-машин достаточно просто. Для этого необходимы сетевая карта Ethernet, установленная на каждой машине, соответствующие кабели Ethernet и другое оборудование. И если в вашей организации или университете есть доступ к Интернету, вы с легкостью можете присоединить вашу систему Linux к этой сети.

Linux поддерживает также SLIP (Serial Line Internet Protocol – интернет-протокол последовательных линий) и PPP (Point-to-Point Protocol – протокол типа

---

[1] Олаф Кирк «Linux для профессионалов. Руководство администратора сети». – Пер. с англ. – СПб.: Питер, 2000.

«точка-точка»). SLIP и PPP позволяют получить доступ к Интернету по коммутируемым телефонным линиям с помощью модема. Если ваша организация или университет предоставляют доступ по SLIP или PPP, можно дозвониться до сервера и соединить вашу машину с Интернетом по телефонной линии. И наоборот, если ваша Linux-система имеет доступ к Интернету через локальную сеть, можно сконфигурировать ее как сервер SLIP или PPP.

В следующих разделах мы не будем касаться SLIP, так как большинство людей в наши дни используют PPP.

Помимо руководства «Linux Network Administrator's Guide» на сайте *http://www.tldp.org/HOWTO/HOWTO-INDEX/networking.html* можно найти большое количество документов, описывающих настройку различных аспектов сетевых взаимодействий, включая инструкции по настройке некоторых аппаратных устройств, таких как модемы. Например, в документе *http://www.tldp.org/HOWTO/Ethernet-HOWTO.html* описана настройка многочисленных драйверов сетевых карт Ethernet.

Представляет интерес и книга «TCP/IP Network Administration»[1], содержащая всю информацию по настройке и использованию TCP/IP в UNIX-системах. Если вы собираетесь построить сеть на Linux-машинах или выполнять любую другую серьезную работу с TCP/IP, следует иметь в запасе знания по сетевому администрированию, представленные в этой книге.

Если вы всерьез решили заняться настройкой и обслуживанием сетей, возможно, вам нужно прочесть книгу «DNS and BIND».[2] В живой и увлекательной манере в ней рассказывается обо всем, что связано с серверами имен.

Основные понятия TCP/IP

Для того чтобы полностью оценить (и использовать) мощные возможности TCP/IP, необходимо поближе познакомиться с его основными принципами. Transmission Control Protocol/Internet Protocol – это набор *протоколов* (это слово будет назойливо звучать в данной главе), определяющий способ взаимодействия различных машин по сети, а также взаимодействие между различными уровнями набора протоколов. Лучшими источниками прочных теоретических знаний в области протоколов Интернета являются книги Дугласа Комера (Douglas Comer) «Internetworking with TCP/IP» (Prentice Hall) и Ричарда Стивенса (W. Richard Stevens) «TCP/IP Illustrated».[3]

TCP/IP был изначально разработан для использования в экспериментальной сети ARPANet с целью поддержки военных и информационных разработок.[4] Поэтому иногда можно услышать, как TCP/IP называют «интернет-протоколами

---

[1] Крэйг Хант «TCP/IP. Сетевое администрирование», 3-е изд. – Пер. с англ. – СПб: Символ-Плюс, 2004.

[2] Крикет Ли и Пол Альбитц «DNS и BIND», 5-е изд. – Пер. с англ. – СПб.: Символ-Плюс, 2008.

[3] У. Стивенс «Протоколы TCP/IP. В подлиннике». – Пер. с англ. – СПб.: BHV-СПб, 2003.

[4] Одно из лучших изложений истории перерастания ARPANet в Интернет находится здесь: *http://www.computer-museum.ru/frgnhist/intern1.htm*. – *Примеч. науч. ред.*

DARPA»[1]. Вслед за этим первым Интернетом возникли другие сети TCP/IP, такие как National Science Foundation's (NSFNET), а также тысячи других региональных и локальных сетей по всему миру. Все эти сети соединены между собой в единый конгломерат, известный как Интернет.

В сети TCP/IP каждой машине присвоен *IP-адрес*, который является 32-разрядным числом, идентифицирующим уникальным образом данную машину. Чтобы организовать сеть и присваивать адреса хостам, нужно немного знать об IP-адресах. IP-адрес обычно представляется в виде четырех десятичных чисел, разделенных точками. Например, IP-адрес 0x80114b14 (в шестнадцатеричном формате) можно записать как 128.17.75.20.

Здесь нужно сказать о двух особых случаях: динамических IP-адресах и маскируемых IP-адресах. Те и другие изобретены, чтобы справиться с возникшей нехваткой IP-адресов (что перестанет быть проблемой, когда все примут новый стандарт IPv6, отводящий 6 байт для IP-адреса – достаточно, чтобы снабдить IP-адресом каждую амебу во Вселенной).[2]

Динамические IP-адреса часто используются в соединениях по коммутируемым линиям: вы дозваниваетесь до провайдера Интернета, который назначает вам IP-адрес из пула, выделенного им для этого вида сервиса. При следующем соединении вы можете получить другой номер IP. Смысл в том, что из всех клиентов провайдера лишь небольшая часть подключается к сети одновременно, поэтому требуется меньшее количество IP-адресов. Тем не менее, пока компьютер подключен к Интернету, его IP-адрес уникален – никакой другой компьютер не использует его в то же самое время.

Маскирование IP-адресов (masquerading, этот прием известен также под названием Network Adress Translation – преобразование сетевых адресов) позволяет пользоваться одним IP-адресом сразу нескольким компьютерам. Все машины в маскируемой сети пользуются локальными IP-адресами, принадлежащими диапазону, который выделен для внутреннего употребления и не может служить реальным адресом в Интернете. Одни и те же локальные IP-адреса могут быть использованы в любом количестве таких сетей, поскольку за пределами локальной сети они никому не видны. Одна машина – сервер маскировки – будет отображать эти закрытые[3] IP-адреса в один открытый IP-адрес (который может быть динамическим или статическим) и с помощью хитроумного механизма отображения направлять входящие пакеты данных на нужную машину.

---

[1] DARPA является прародителем TCP/IP, который существенно сложнее DARPA и позже заменил этот ранний протокол. – *Примеч. науч. ред.*

[2] В Linux уже имеется поддержка протокола IPv6, но, к сожалению, большинство локальных сетей и провайдеров доступа в Интернет еще не используют его, поэтому подробно рассказывать о нем было бы несколько преждевременно.

[3] Важно не только то, что эти адреса общие для разных LAN, но и то, что они при правильном применении выбираются из диапазонов локальных адресов, определенных стандартом TCP/IP: 10.*.*.* для сетей класса A; 172.16.*.*–172.32.*.* для сетей класса B; 192.168.*.* – для сетей класса C; согласно стандарту, если пакет с одним из локальных адресов поступит на маршрутизатор, тот обязан отбросить его и не ретранслировать далее. Таким образом, пакет с локальным исходящим адресом не может покинуть LAN, в которой он порожден. – *Примеч. науч. ред.*

IP-адрес делится на две части: адрес сети и адрес хоста. Старшие биты адреса указывают на адрес сети, а оставшиеся биты определяют адрес хоста. (Обычно каждый *хост*, или узел сети, является отдельной машиной в сети.) Размер этих двух полей адреса зависит от типа рассматриваемой сети. Например, в сети класса B (где первый байт IP-адреса имеет значение от 128 до 191) два первых байта идентифицируют сеть, а два оставшихся байта идентифицируют хост (рис. 13.1). В приведенном выше примере IP-адреса сеть имеет адрес 128.17, а адрес хоста – 75.20. Другими словами, машина с IP-адресом 128.17.75.20 является хостом с номером 75.20 в сети 128.17.

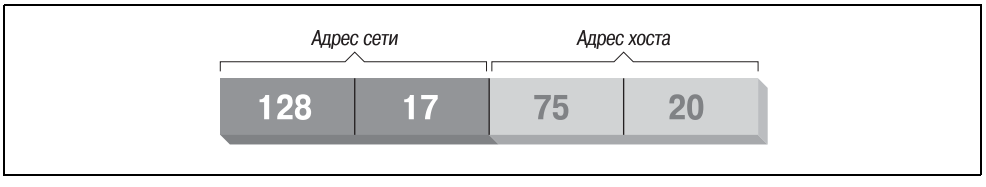


Рис. 13.1. *IP-адрес*

Кроме того, часть IP-адреса, идентифицирующая хост, может быть разделена далее для выделения *адреса подсети*. Такое разбиение на подсети позволяет разделять большие сети на более мелкие подсети, каждой из которых можно управлять независимо от других сетей. Например, если организации выделена одна сеть класса B, в которой под информацию о хосте используется 2 байта, то в такой сети может быть до 65 534 хостов.[1] При желании сеть может быть разделена на подсети, управляемые разными отделами данной организации. При разделении на подсети первый байт из адреса хоста (который является третьим байтом во всем IP-адресе) может использоваться в качестве адреса подсети, а второй байт будет считаться адресом хоста для этой подсети (рис. 13.2). В этом случае IP-адрес 128.17.75.20 идентифицирует хост номер 20 в подсети 75 сети 128.17.[2]

Процессы (на одной или на разных машинах), которым необходима связь по TCP/IP, обычно указывают IP-адрес нужной машины, а также *номер порта*. IP-адрес используется, конечно, для определения маршрута передачи данных вызываемой машине. Номер порта является 16-разрядным числом, определяющим

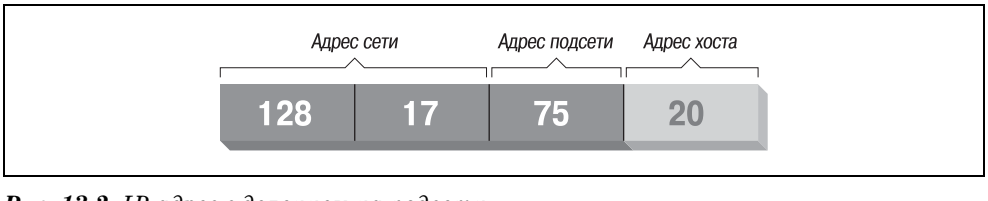


Рис. 13.2. *IP-адрес с делением на подсети*

[1] Почему не 65 536? По причинам, которые мы обсудим позже, адреса 0 и 255 недопустимы.

[2] Здесь приведен частный пример. В общем случае граница, разделяющая поля подсети и хоста, определяется маской подсети, являющейся такой же индивидуальной характеристикой каждого сетевого интерфейса, как и сам IP-адрес. – *Примеч. науч. ред.*

приложение или службу на вызываемой машине, которой предназначены данные. Номера портов можно представить как номера офисов в большом офисном здании: все здание имеет один IP-адрес, но у каждой фирмы в этом здании есть отдельный офис со своим номером.

Рассмотрим пример реального использования IP-адресов и номеров портов. Программа `ssh` предоставляет возможность пользователю одной машины начать сеанс регистрации на другой машине, причем все данные, которыми обмениваются эти две машины, будут зашифрованы. На удаленной машине работает демон *ssh – sshd*, который прослушивает определенный порт, ожидая входящие соединения (в данном случае порт имеет номер 22).[1]

При запуске *ssh* пользователь указывает адрес системы, к которой он хочет подключиться, и программа *ssh* пытается открыть соединение с портом 22 на удаленной машине. В случае успеха *ssh* и *sshd* получают возможность общаться между собой, чтобы предоставить вход данному пользователю.

Обратите внимание, что *ssh*-клиент на локальной машине тоже имеет свой номер порта. Этот номер динамически присваивается клиенту при его запуске, поскольку удаленному серверу *sshd* нет необходимости заранее знать номер порта входящего клиента *ssh*. Когда клиент инициирует соединение, номер его порта является частью информации, посылаемой демону *sshd*. Сервер *sshd* можно представить как фирму с известным всем адресом. Любой, кто хочет связаться с *sshd*, запущенным на определенной машине, должен знать не только IP-адрес машины (адрес офисного здания), но и номер порта (номер офиса в здании), где можно будет найти *sshd*. Адрес и номер порта *ssh*-клиента включены как часть «обратного адреса» на конверте.

Семейство TCP/IP содержит несколько протоколов. Transmission Control Protocol (TCP), или протокол управления передачей, отвечает за надежную связь с установкой соединения между двумя процессами, которые могут быть запущены на разных машинах в сети. User Datagram Protocol (UDP), или протокол пользовательских дейтаграмм, похож на TCP, за исключением того, что он не устанавливает соединение и не гарантирует доставку пакетов. Процессы, использующие UDP, должны иметь собственные процедуры подтверждения и синхронизации, если это необходимо.

TCP и UDP передают и получают данные блоками, которые называют *пакетами*. Каждый пакет содержит фрагмент информации, которую нужно передать другой машине, а также заголовок, определяющий адреса портов отправителя и получателя.

Internet Protocol (IP), или межсетевой протокол, находится в иерархии протоколов ниже TCP и UDP. Он отвечает за передачу и маршрутизацию пакетов TCP или UDP по сети. Для этого протокол IP заключает каждый пакет TCP или UDP в другой пакет (известный как *дейтаграмма* IP), который содержит заголовок с информацией о маршруте и получателе. В заголовке дейтаграммы IP содержатся IP-адреса машин отправителя и получателя.

---

[1] На многих системах *sshd* не всегда прослушивает порт 22, а вместо него этот порт прослушивает демон служб Интернета – `inetd`. Но сейчас мы не будем отвлекаться на такие детали.

Следует отметить, что IP ничего не знает о номерах портов; за их интерпретацию отвечают TCP и UDP. Аналогично TCP и UDP не обрабатывают IP-адреса, которые находятся (как видно из названия) исключительно в ведении IP. Как видите, метафора с почтой, конвертами и адресами подходит достаточно точно: каждый пакет можно представить в виде письма, вложенного в конверт. TCP и UDP запечатывают письмо в конверт, на котором указывают номера портов отправителя и адресата (номер офиса).

IP выступает в роли почтовой службы офисного здания. IP получает конверт и запечатывает его в другой конверт с IP-адресами получателя и отправителя (адрес офисного здания). Служба доставки (о которой мы еще не говорили) доставляет письмо в соответствующее офисное здание. Здесь почтовая служба распечатывает наружный конверт и передает письмо протоколу TCP или UDP, которые доставляют его в соответствующий офис, основываясь на номере порта (он написан на внутреннем конверте). Каждый конверт имеет обратный адрес, который IP и TCP/UDP используют для ответа на письмо.

Чтобы указывать машины в Интернете более понятным человеку способом, хостам наряду с IP-адресами даются имена. Служба доменных имен (Domain Name Service, DNS) заботится о преобразовании имен хостов в IP-адреса и наоборот, а также управляет распределением в рамках всего Интернета базы данных с соответствиями имен и IP-адресов. Использование имен хостов также позволяет изменять IP-адрес, связанный с машиной (например, если компьютер перевели в другую сеть), не беспокоясь о том, что после смены адреса машину нельзя будет найти. Необходимо лишь обновить запись в DNS для этого хоста, и все ссылки на эту машину по имени будут работать и дальше.[1]

DNS является огромной, распределенной по всему миру базой данных. Каждая организация управляет частью этой базы, в которой перечислены имеющиеся в ней машины. Если в вашей организации эта работа поручена вам, помощь можно получить из книги «Linux Network Administrator's Guide» или «TCP/IP. Network Administration». Если этого недостаточно, наиболее полную информацию вы найдете в книге «DNS and BIND».

Для целей обычного администрирования достаточно знать, что либо в вашей системе должен быть запущен демон *named* (произносится как «name-dee»), либо система должна быть настроена на использование службы доменных имен, которая обычно предоставляется поставщиком интернет-услуг или имеется в локальной сети. Этот демон является вашим окном в DNS.

Может возникнуть вопрос, как пакет от одной машины (офисное здание) добирается до другой. Это главная задача IP, а также некоторых других протоколов, которые помогают IP в его работе. Кроме обработки IP-дейтаграмм на каждом хосте (как на почте) IP отвечает за маршрутизацию пакетов между хостами.

---

[1] IP-адрес, таким образом, является инженерным понятием, техническим адресом, на который IP-сеть всегда перенаправит пакет, а DNS-адрес – только ссылкой «справочной службы». При технической реконструкции и изменениях IP-адресов узлов пакеты будут корректно доставляться при обращении по DNS-адресу и его разрешению в IP (о чем сказано выше). С другой стороны, при перебоях в работе служб DNS (а такое случается) отправка с использованием символьного адреса становится невозможной, но отправка с указанием прямого численного IP-адреса того же хоста в это же время прекрасно работает. – *Примеч. науч. ред.*

Прежде чем обсуждать, как работает маршрутизация, мы должны описать модель, по которой построены сети TCP/IP. Сеть – это просто набор машин, которые физически некоторым образом соединены между собой, например при помощи Ethernet или последовательных соединений. В терминах TCP/IP каждая сеть имеет свои собственные методы внутренней маршрутизации и передачи пакетов.

Сети соединены между собой *шлюзами* (*gateway*), которые также известны как *маршрутизаторы* (*routers*). Шлюз – это хост, имеющий прямое соединение с двумя или более сетями; он может передавать информацию между сетями и направлять пакеты из одной сети в другую. В частности, шлюзом может быть рабочая станция, имеющая более одного интерфейса Ethernet. Каждый из интерфейсов соединен со своей сетью, и операционная система использует эти соединения для того, чтобы машина могла работать в качестве шлюза.

Чтобы конкретизировать наши рассуждения, представим себе воображаемую сеть, состоящую из машин *eggplant*, *papaya*, *apricot* и *zucchini*. На рис. 13.3 представлена конфигурация этих машин в сети.

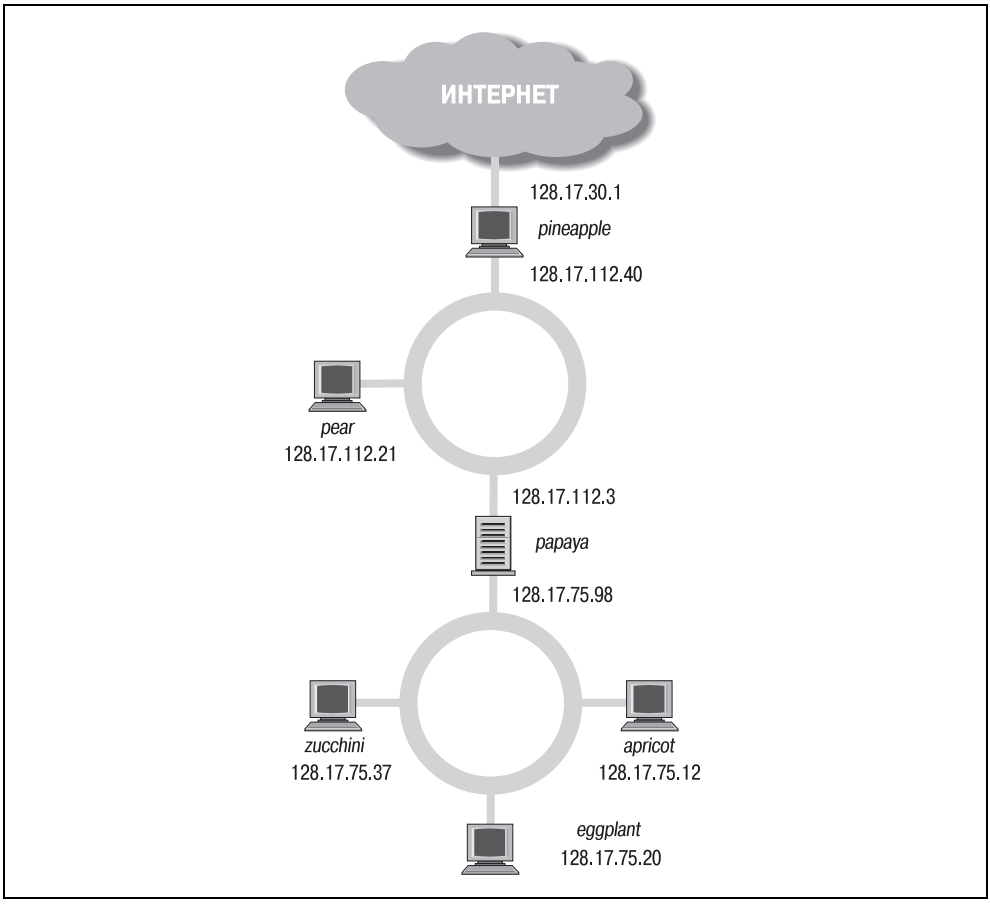


Рис. 13.3. *Сеть с двумя шлюзами*

Как можно видеть, у *papaya* два IP-адреса: один в подсети 128.17.75 и другой в подсети 128.17.112. Два IP-адреса есть и у *pineapple*: один для подсети 128.17.112, а другой для подсети 128.17.30.

Чтобы определить маршрут передачи пакетов между машинами, IP использует сетевую часть IP-адреса. Для этого на каждой машине в сети имеется *таблица маршрутизации* (*routing table*), в которой хранится список сетей и шлюзов для этих сетей. Для доставки пакета на определенную машину IP просматривает сетевую часть адреса назначения. Если для этой сети есть запись в таблице маршрутизации, IP отправляет пакет через соответствующий шлюз. В противном случае пакет отправляется через шлюз по умолчанию, заданный в таблице.

Таблицы маршрутизации могут содержать записи как для сетей, так и для отдельных машин. Кроме того, на каждой машине есть запись о маршруте для себя самой. Рассмотрим таблицу маршрутизации для машины *eggplant*. Выполнив команду *netstat -rn*, мы увидим следующее:

```
eggplant:$ netstat -rn
Kernel IP routing table
Destination   Gateway        Genmask         Flags  MSS   Window irtt  Iface
128.17.75.0   128.17.75.20   255.255.255.0   UN     1500  0          0  eth0
default       128.17.75.98   0.0.0.0         UGN    1500  0          0  eth0
127.0.0.1     127.0.0.1      255.0.0.0       UH     3584  0          0  lo
128.17.75.20  127.0.0.1      255.255.255.0   UH     3584  0          0  lo
```

Первая колонка содержит адреса сетей (и хостов) назначения, включенных в таблицу. Первой указана запись для сети 128.17.75 (заметьте: в записях для сетей адрес хоста равен 0), где расположен хост *eggplant*. Все пакеты, посылаемые в эту сеть, будут отправлены через шлюз 128.17.75.20, который является IP-адресом *eggplant*. Обычно маршрут к собственной сети идет через интерфейс самой машины.

Колонка `Flags` таблицы маршрутизации дает информацию об адресе назначения для этой записи. `U` указывает на то, что маршрут работает («up»), `N` указывает на то, что это маршрут к сети, `H` – на то, что это маршрут к хосту, и т. д. Поле `MSS` показывает число байтов, которое может быть отправлено соответствующему соединению за один раз (максимальный размер сегмента), `Window` указывает количество фреймов, которое может быть отправлено до получения подтверждения о приеме, `irtt` выдает статистику использования маршрута, а `Iface` указывает сетевой интерфейс, используемый для этого маршрута. В системах Linux интерфейсы Ethernet именуются как *eth0*, *eth1* и т. д. *lo* – это петлевое устройство (loopback), о котором мы вскоре расскажем.

Вторая запись в таблице – это маршрут по умолчанию, применяемый ко всем пакетам, посылаемым сетям или отдельным хостам, не имеющим записи в таблице. Здесь маршрут по умолчанию лежит через машину *papaya*, которую можно считать дверью во внешний мир. Все машины в подсети 128.17.75 для связи с машинами любой другой подсети должны пройти через машину *papaya*.

Третья запись в таблице сделана для адреса 127.0.0.1, который является адресом *петлевого* (*loopback*) интерфейса. Этот адрес используется, если машине необходимо выполнить подключение по TCP/IP к себе самой. Устройство *lo* используется в качестве интерфейса с целью предотвращения использования сети Ethernet (через интерфейс *eth0*) при петлевых подключениях. В этом случае ресурсы сети не тратятся впустую, когда компьютер хочет поговорить сам с собой.

Последняя запись в таблице маршрутизации сделана для IP-адреса 128.17.75.20, который является собственным адресом хоста *eggplant*. Как можно заметить, в качестве шлюза выступает адрес 127.0.0.1. Таким образом, каждый раз, когда машина *eggplant* создает подключение по TCP/IP к самой себе, она делает это через сетевое «устройство» *lo*, а адрес петлевого интерфейса используется как шлюз.

Представим себе, что *eggplant* хочет послать пакет *zucchini*. Дейтаграмма IP содержит адрес источника 128.17.75.20 и адрес назначения 128.17.75.37. Протокол IP определяет, что сетевая часть адреса назначения равна 128.17.75, и, соответственно, использует запись для 128.17.75.0 в таблице маршрутизации. Пакет посылается прямо в сеть, где *zucchini* сможет получить его и обработать.

Что произойдет, если *eggplant* захочет послать пакеты машине, находящейся вне локальной сети, например *pear*? В этом случае адрес назначения будет 128.17.112.21. Протокол IP попытается найти в таблице маршрутизации маршрут для сети 128.17.112, но поскольку такого маршрута в таблице нет, будет выбран шлюз по умолчанию, которым является *papaya*. Получив пакет, *papaya* отыщет адрес назначения в своих собственных таблицах маршрутизации. Таблица маршрутизации на машине *papaya* может выглядеть следующим образом:

```
Destination   Gateway        Genmask         Flags   MSS   Window   irtt   Iface
128.17.75.0   128.17.75.98   255.255.255.0   UN      1500  0        0      eth0
128.17.112.0  128.17.112.3   255.255.255.0   UN      1500  0        0      eth1
default       128.17.112.40  0.0.0.0         UGN     1500  0        0      eth1
127.0.0.1     127.0.0.1      255.0.0.0       UH      3584  0        0      lo
128.17.75.98  127.0.0.1      255.255.255.0   UH      3584  0        0      lo
```

Мы видим, что *papaya* соединена с сетью 128.17.75 через устройство *eth0* и с сетью 128.17.112 – через устройство *eth1*. Маршрут по умолчанию идет через машину *pineapple*, которая является шлюзом в Большое Нечто (насколько это известно машине *papaya*).

Получив пакет, предназначенный для *pear*, маршрутизатор *papaya* видит, что адрес назначения находится в сети 128.17.112, и отправляет этот пакет, используя вторую запись в таблице маршрутизации.

Аналогично, если *eggplant* хочет связаться с машинами, находящимися за пределами локальной сети организации, она отправляет пакет через шлюз *papaya*. Оттуда, в свою очередь, исходящие пакеты будут отправлены через *pineapple* и далее. Пакеты передаются от одного шлюза другому до тех пор, пока они не достигнут сети назначения. Такова структура, на которой основан Интернет: кажущаяся бесконечной цепь сетей, связанных между собой шлюзами.[1]

---

[1] Выше описана система «необходимых и достаточных» правил для маршрутизации любого пакета в TCP/IP: а) если адрес хоста/подсети известен из таблицы маршрутизации для какого-то сетевого интерфейса (первого найденного, если таких несколько), то пакет направляется через этот интерфейс; б) если ни для какого хоста соответствие не найдено, пакет отправляется через интерфейс по умолчанию. Для «достаточности» должно быть добавлено третье правило: в) пакет, поступивший хосту извне через некоторый интерфейс, не может быть отправлен через этот же интерфейс и уничтожается (в противном случае возникали бы бесконечные петли в сети и «размножение» пакетов). Детальнее см. в книге У. Стивенса «Протоколы TCP/IP. В подлиннике», BHV-СПб, 2003. – *Примеч. науч. ред.*

Требования к аппаратуре

Протоколами TCP/IP можно пользоваться в Linux без какого-либо сетевого оборудования вообще, настроив петлевой режим («loopback»), который позволяет машине подключаться к себе самой. Это необходимо для некоторых приложений и игр, использующих сетевое «устройство» петлевого интерфейса.

Однако, если вы хотите использовать Linux в сети Ethernet TCP/IP, очевидно, вам потребуется сетевой адаптер Ethernet. Linux поддерживает множество адаптеров для шин ISA, EISA, PCI, а также USB и PCMCIA. Полную информацию по аппаратной совместимости с устройствами Ethernet вы найдете в документе «Linux Ethernet HOWTO». В любом достаточно новом компьютере (который был продан в последние, скажем, два-три года), скорее всего, имеется встроенный адаптер Ethernet, благодаря чему отпадает необходимость установки дополнительной платы адаптера Ethernet. Определить наличие встроенного адаптера можно по гнезду Ethernet (например, RJ45) где-нибудь на корпусе системного блока.

В последние годы в Linux была добавлена поддержка высокоскоростных сетевых соединений, отличных от Ethernet, таких как HIPPI. Эта тема выходит за рамки нашей книги, но если она вас интересует, некоторую информацию можно получить в каталоге *Documentation/networking* исходного кода ядра.

Если у вас есть соединение ADSL и используется маршрутизатор ADSL, то для Linux оно представляется обычным соединением Ethernet, поэтому не требуется ни специального оборудования (кроме карты Ethernet, разумеется), ни особых драйверов помимо драйвера самой карты Ethernet. Если вы хотите соединить машину Linux непосредственно с модемом ADSL, то вам также не потребуется специального оборудования или драйвера, но нужно будет запустить протокол под названием PPPoE (PPP over Ethernet), о чем более подробно будет сказано далее.

Linux также поддерживает протоколы SLIP и PPP, которые позволяют использовать модемы для связи с Интернетом по телефонным линиям. В этом случае вам потребуется модем, совместимый с сервером SLIP или PPP, к которому вы подключаетесь. Например, для многих серверов требуются модемы 56kbps V.90 (большинство также поддерживают K56flex). В этой книге мы опишем настройку PPP, так как именно этот протокол используется большинством поставщиков услуг доступа в Интернет.

Наконец, существует протокол PLIP, позволяющий соединить два компьютера через их параллельные порты, для чего требуется специальный соединительный кабель.

Настройка TCP/IP для работы в сети Ethernet

В этом разделе мы обсудим порядок настройки TCP/IP в системах Linux для подключения к Ethernet. Возможно, эта система будет частью локальной сети, в которой уже используется TCP/IP. В этом случае уже будут настроены и доступны шлюз, сервер имен и т. д.

Следующая информация касается в первую очередь подключения к Ethernet. Если вы собираетесь использовать PPP, прочтите этот раздел, чтобы познакомиться с основными понятиями, а специальные инструкции по настройке PPP вы найдете в разделе «Коммутируемые линии и протокол PPP» далее в этой главе.

С другой стороны, вы, возможно, хотите установить целую сеть из машин Linux (или из машин Linux и других систем). В этом случае вам придется позаботиться о ряде вопросов, которые здесь не обсуждаются. Сюда входит установка своего сервера имен, а также настройка шлюза, если вашу сеть необходимо соединить с другими сетями. Если вашу сеть необходимо соединить с Интернетом, нужно также получить IP-адреса и сопутствующую информацию у вашего провайдера.

Описанный здесь метод должен работать во многих системах Linux, настроенных для работы в уже имеющихся сетях, но наверняка не для всех. За более подробной информацией мы направляем вас к книгам по сетевому администрированию TCP/IP, которые были упомянуты в начале этой главы.

Прежде всего, мы будем исходить из предположения, что на вашей системе Linux установлено все необходимое программное обеспечение TCP/IP. В него входят основные клиенты, такие как *ssh* и FTP, команды системного администрирования, такие как *ifconfig* и *route* (обычно расположенные в каталогах */etc* или */sbin*), и файлы настройки сети (такие как */etc/hosts*). Документы по сетям в Linux, о которых говорилось ранее, помогут разобраться с установкой поддержки сети в Linux, если у вас ее еще нет.

С целью унификации решения различных задач, связанных с настройкой сетей и описанием маршрутов, был разработан новый интерфейс системного администрирования, представленный единственной командой `ip`, которая распространяется в составе пакета IPROUTE2. Мы не будем описывать здесь эту команду, потому что основное ее предназначение находится в области настройки дополнительных возможностей, которые большинством администраторов не используются, но как только вы разберетесь с основными понятиями, о которых говорится в данной главе, вы без труда сможете самостоятельно изучить порядок работы с данной командой.

Мы также будем предполагать, что ядро было настроено и скомпилировано с включенной поддержкой TCP/IP.[1] Информация по сборке ядра системы приводится в разделе «Сборка нового ядра» главы 18. Чтобы активировать сетевые функции в ядре, необходимо утвердительно ответить на соответствующие вопросы на этапе *make config* или *make menuconfig*, затем пересобрать ядро и перезагрузить машину.

Затем необходимо будет изменить ряд файлов с настройками. По большей части это простая процедура. Однако, к сожалению, между различными производителями дистрибутивов Linux имеются значительные разногласия по поводу того, где следует находиться многочисленным программам и файлам настройки TCP/IP. В большинстве случаев их можно найти в каталоге */etc*, но иногда их можно обнаружить в */usr/etc*, */usr/etc/inet* или в других необычных местах. В худшем случае вам придется прибегнуть к помощи команды `find`, чтобы отыскать эти файлы. Кроме того, имейте в виду, что не во всех дистрибутивах файлы с настройками и программное обеспечение находятся в одном месте; они могут быть разнесены по разным каталогам.

Мы расскажем о том, как установить и настроить поддержку сети в Linux вручную. В результате вы должны получить некоторое представление о том, что

[1] Обычно в любом дистрибутиве ядро, собранное для установки по умолчанию, уже имеет поддержку TCP/IP. – *Примеч. науч. ред.*

в действительности происходит, и самостоятельно справиться с проблемами, которые могут возникнуть при автоматической установке, осуществляемой вашим дистрибутивом. Однако разумно попробовать сначала установить поддержку сети с помощью тех программ настройки, которые есть в вашем дистрибутиве. Многие из них стали достаточно развитыми и могут автоматически определять ряд необходимых настроек. Как правило, все эти программы доступны из главного меню рабочего стола. Если вы до конца поймете основные принципы настройки сетей, излагаемые в этой главе, вы без труда разберетесь с тем, как этими программами пользоваться. Мы же, в свою очередь, не будем заниматься здесь описанием этих программ, потому что они, как правило, очень быстро меняются, и потому самое главное, что вы должны знать, – это окончательная цель, которой вы стремитесь достичь.

В этом разделе предполагается, что в системе используется одно устройство Ethernet. Если в вашей системе имеется более одного сетевого соединения (следовательно, система выступает в роли шлюза), все указания могут быть с легкостью экстраполированы на большее число сетевых интерфейсов.

Здесь мы также обсудим настройку систем с единственным петлевым интерфейсом (без Ethernet или PPP-соединений). Если у вас нет доступа к сети, вы, возможно, захотите настроить TCP/IP на использование только петлевого интерфейса (loopback). Тогда вы сможете без реального подключения к сети использовать приложения, которые требуют TCP/IP.

Настройка сети

Прежде чем вы начнете настраивать TCP/IP, необходимо выяснить следующую информацию о параметрах вашей сети. В большинстве случаев эту информацию может предоставить администратор локальной сети или провайдер доступа к сети. Если в вашей сети используется протокол динамического назначения IP-адресов (DHCP), порядок настройки будет выглядеть несколько иначе, например, вам не нужно будет заранее знать свой IP-адрес, так как он будет назначаться автоматически. Но зачастую изучение принципов настройки лучше выполнять небольшими шагами, и начать следует со случая статического IP-адреса. Для этого достаточно убедиться вместе с сетевым администратором или поставщиком интернет-услуг, что вы не используете уже занятый кем-либо IP-адрес, так как в противном случае вы и пользователь другой системы можете испытать немало неприятных минут, потому что нормальная работа в сети будет нарушена, если не сказать хуже.

Ваш IP-адрес

Это уникальный адрес машины в формате десятичных чисел, разделенных точками, например 128.17.75.98. Этот номер предоставляется вам сетевым администратором.

Если вы настраиваете петлевой интерфейс (в этом случае нет PPP и сетевой карты, а имеется только TCP/IP-подключение к собственной машине), вашим IP-адресом будет 127.0.0.1.[1]

[1] Или по желанию любой из адресов диапазона 127.0.0.*. – *Примеч. науч. ред.*

Маска подсети

Это четыре числа, разделенные точками, как IP-адрес, определяющие, какая часть IP-адреса является адресом подсети, а какая – адресом хоста в этой подсети.

Маска подсети является числом, которое складывается с IP-адресом при помощи операции логического И, в результате чего выясняется, к какой подсети принадлежит ваш адрес. Например, ваша маска подсети может быть такой: 255.255.255.0. Если ваш IP-адрес – 128.17.75.20, то часть адреса, указывающая на подсеть, будет 128.17.75.

Следует различать «адрес сети» и «адрес подсети». Напоминаем, что для адресов сетей класса B первые два байта (в данном случае 128.17) определяют сеть, а вторые два байта указывают на хост. Однако с маской подсети 255.255.255.0 адрес 128.17.75 считается адресом целой подсети (то есть подсеть 75 сети 128.17), а 20 будет адресом хоста.

Сетевую маску выбирает сетевой администратор, и поэтому он может предоставить вам эту информацию. То же касается и петлевого интерфейса. Поскольку его адрес всегда равен 127.0.0.1, сетевая маска для этого адреса всегда будет равна 255.0.0.0.

Адрес вашей подсети

Это часть вашего IP-адреса, указывающая на подсеть, как это определено маской подсети. Например, при маске подсети 255.255.255.0 и IP-адресе 128.17.75.20 адрес подсети будет 128.17.75.0.

Системы, обладающие только петлевым интерфейсом, не имеют адреса подсети.

Широковещательный адрес

Этот адрес используется для отправки широковещательных пакетов всем машинам вашей подсети. Обычно он равен адресу подсети (о котором говорилось в предыдущем пункте), где на месте хоста стоит 255. Для подсети с адресом 128.17.75.0 широковещательный адрес будет 128.17.75.255. Аналогично для подсети с адресом 128.17.0.0 широковещательный адрес будет 128.17.255.255.

Обратите внимание: некоторые системы используют сам адрес подсети для отправки широковещательных сообщений. Если у вас есть какие-нибудь сомнения, поделитесь ими с администратором вашей сети.

Системы, обладающие только петлевым интерфейсом, не имеют широковещательного адреса.

IP-адрес вашего шлюза

Это адрес машины, которая выступает шлюзом по умолчанию для связи с внешним миром. На самом деле у вас может быть несколько адресов шлюзов, например, если ваша сеть напрямую соединена с несколькими другими сетями. Однако только один из шлюзов будет выступать в качестве маршрута *по умолчанию*. (Вспомните пример из предыдущего раздела, где сеть 128.17.112.0 соединена через *papaya* с сетью 128.17.75.0 и выходит в остальной мир через *pineapple*.)

Ваши сетевые администраторы предоставят информацию об IP-адресах всех шлюзов вашей сети, а также о сетях, с которыми они соединяют вашу сеть.

Позже вы используете эти данные, когда с помощью команды `route` будете вводить в таблицу маршрутизации записи для каждого шлюза.

Системы, обладающие только петлевым интерфейсом, не имеют адресов шлюзов. То же относится и к изолированным сетям.[1]

IP-адрес вашего сервера имен

Это адрес машины, выполняющей преобразование имен хостов в адреса для вашей системы. Этот адрес вам предоставят администраторы сети.

Возможно, вы захотите запустить собственный сервер имен (настроив и запустив `named`). Однако, если нет крайней необходимости в собственном сервере имен[2] (например, если другого в вашей локальной сети нет), мы рекомендуем использовать адрес сервера имен, который вам предоставил администратор. Информация по управлению сервисом `named` содержится в большинстве книг по настройке TCP/IP.

Естественно, системы, имеющие только петлевой интерфейс, не имеют адреса сервера имен.

Файлы с настройками сети

Файлы с настройками сети – это системные сценарии, запускаемые *init* во время начальной загрузки и выполняющие настройку сетевого окружения. Они запускают все основные системные демоны (такие как *sendmail*, *crond* и т. д.) и используются для настройки параметров сетей. Файлы *rc* обычно находятся в каталоге */etc/init.d*.

Обратите внимание: существует *масса* различных способов управлять настройками сети, описанными здесь. Каждый дистрибутив Linux использует слегка отличающийся от других механизм автоматизации этих настроек. Описываемый нами метод является универсальным и позволяет ограничиться созданием всего двух файлов с настройками, которые выполнят все действия, необходимые для выхода вашей машины в сеть. В большинстве дистрибутивов есть свои собственные сценарии, которые выполняют примерно то же самое. Если у вас есть сомнения, сначала попытайтесь настроить сеть так, как предлагается в документации к вашему дистрибутиву, и лишь в качестве последнего средства используйте методы, описываемые здесь. (Например, дистрибутив Red Hat использует сценарий */etc/rc.d/init.d/network*, который получает информацию о сети из файлов, находящихся в каталоге */etc/sysconfig*. Программа `control-panel`, поставляемая вместе с Red Hat, выполняет настройку сети автоматически без редактирования этих файлов. Дистрибутив SUSE использует файл */sbin/init.d/network* и позволяет настроить большую часть параметров сетевого окружения с помощью ути-

---

[1] Изолированные LAN не имеют шлюза по умолчанию в общепринятом смысле: им некуда отсылать пакеты вне сети. Но и в этом случае лучше установить на хосте значение IP-шлюза по умолчанию – любого хоста LAN или даже произвольный IP-адрес, так как это может иметь значение при использовании адресов групповой рассылки и протокола IGP. – *Примеч. науч. ред.*

[2] Считается, что устанавливать локальный сервер DNS имеет смысл только в режиме «кэширующего сервера»; такой режим можно обеспечить и с `named`. – *Примеч. науч. ред.*

литы *YaST2*. И уж конечно, любая новая версия любого дистрибутива может выполнять те же действия совершенно отличным способом.)

Далее мы опишем файлы с настройками, используемые для определения параметров TCP/IP в наиболее распространенных дистрибутивах:

Red Hat

Сетевые настройки разбросаны по файлам для каждого уровня *init*, который предполагает работу с сетью. В частности, каталог */etc/rc.d/rc1.d* управляет загрузкой уровня 1 (однопользовательский режим), и поэтому там нет никаких сетевых команд, а */etc/rc.d/rc3.d* управляет загрузкой уровня 3 и имеет специальные файлы для запуска сети.

SUSE

Все файлы запуска системных сервисов сгруппированы в одном каталоге */sbin/init.d*. Они достаточно универсальны и получают фактические значения из системного файла с настройками */etc/sysconfig*. Самыми важными файлами здесь являются: */etc/init.d/network*, который запускает и останавливает сетевые интерфейсы и настраивает таблицы маршрутизации, и */sbin/init.d/setserial*, который выполняет настройку параметров последовательных портов. Если у вас есть оборудование ISDN, для его настройки используется сценарий */etc/init.d/isdn*. Имейте в виду, что обычно не требуется (да и не следует) редактировать эти файлы, все необходимые изменения следует вносить в файл */etc/sysconfig*.

Debian

Настройка сети (сетевые карты, IP-адреса и таблица маршрутов) и запуск основных сетевых демонов (`portman` и `inetd`) выполняются в файле */etc/init.d/networking*.

Для иллюстрации мы возьмем два файла: */etc/init.d/rc.inet1* и */etc/init.d/rc.inet2*. Первый используется для настройки оборудования и основных сетевых параметров, а второй настраивает сетевые службы. Такое разделение используется во многих дистрибутивах, хотя имена файлов могут различаться.

init использует файл */etc/inittab*, чтобы выбрать процесс, который надо запустить во время загрузки. Для запуска файлов */etc/init.d/rc.inet1* и */etc/init.d/rc.inet2* из *init* в файле */etc/inittab* могут содержаться следующие строки:

```
n1:34:wait:/etc/init.d/rc.inet1
n2:34:wait:/etc/init.d/rc.inet2
```

Файл *inittab* будет описан в разделе «Файлы init, inittab и rc» главы 17. Первое поле задает уникальный двухсимвольный идентификатор для каждой записи. Во втором поле перечислены уровни исполнения, на которых запускаются сценарии; в данной системе сеть инициализируется на уровнях запуска 3 и 4. Слово `wait` в третьем поле сообщает *init*, что продолжать работу можно только после завершения выполнения сценария.

В процессе выполнения настройки сети, возможно, появится необходимость запустить *rc.inet1* и *rc.inet2* вручную (обладая привилегиями пользователя *root*) с целью отладки каких-либо проблем. Позднее можно добавить записи для них в другой *rc*-файл или в */etc/inittab*.

Как уже упоминалось ранее, *rc.inet1* производит настройку сетевого интерфейса, устанавливая IP-адрес и адрес сети, а также заполняет таблицу маршрутизации. Для настройки этих параметров используются две программы: *ifconfig* и *route*, которые обычно находятся в каталоге */sbin*.

ifconfig используется для настройки определенных параметров сетевого интерфейса, таких как IP-адрес, маска подсети, широковещательный адрес и т. д. Для создания и модификации таблицы маршрутизации используется `route`.

В большинстве случаев подойдет файл *rc.inet1*, подобный приведенному ниже. Разумеется, придется отредактировать его для использования в своей системе. Не используйте IP-адрес и адрес сети, приведенные здесь в качестве примера, так как они могут совпадать с адресом реальной машины в Интернете:

```
#!/bin/sh
# Файл /etc/init.d/rc.inet1 - настройка TCP/IP интерфейсов.
# В первую очередь выполняется настройка петлевого (loopback) интерфейса.

HOSTNAME=`hostname`

/sbin/ifconfig lo 127.0.0.1 # по умолчанию используется маска сети 255.0.0.0
/sbin/route add 127.0.0.1   # маршрут к устройству loopback

# Затем настраиваются адаптеры Ethernet. Если вы используете
# только loopback или SLIP, закомментируйте остальные строки.

# Измените значения параметров в соответствии с вашими настройками.
IPADDR="128.17.75.20"      # ЗАМЕНИТЬ своим IP-адресом
NETMASK="255.255.255.0"    # ЗАМЕНИТЬ своей маской подсети
NETWORK="128.17.75.0"      # ЗАМЕНИТЬ своим адресом сети
BROADCAST="128.17.75.255"  # ЗАМЕНИТЬ своим широковещательным адресом
GATEWAY="128.17.75.98"     # ЗАМЕНИТЬ адресом своего шлюза по умолчанию

# Настройка устройства eth0 с использованием вышеуказанных параметров.
/sbin/ifconfig eth0 ${IPADDR} netmask ${NETMASK} broadcast ${BROADCAST}

# Добавить маршрут для собственной подсети.
/sbin/route add ${NETWORK}

# Добавить маршрут для шлюза по умолчанию.
/sbin/route add default gw ${GATEWAY} metric 1

# Конец настройки Ethernet
```

Как вы видели, команда *ifconfig* имеет следующий формат:

```
ifconfig interface device options...
```

Например:

```
ifconfig lo 127.0.0.1
```

присваивает устройству *lo* (loopback) IP-адрес 127.0.0.1, а команда:

```
ifconfig eth0 127.17.75.20
```

присваивает устройству *eth0* (первая карта Ethernet) адрес 127.17.75.20.

Помимо указания IP-адреса устройства Ethernet обычно требуют, чтобы были установлены маска подсети с помощью параметра *netmask* и широковещательный адрес с помощью параметра *broadcast*.

Команда *route* имеет здесь следующий формат:

```
route add [ -net | -host ] destination [ gw gateway ]
[ metric metric ] options
```

где *destination* – это адрес назначения для данного маршрута (либо ключевое слово `default`), *gateway* – IP-адрес шлюза для данного маршрута и *metric* – метрика маршрута (которую мы обсудим позже).

Для добавления записей в таблицу маршрутизации используется команда *route*. Необходимо добавить маршрут для петлевого интерфейса (как говорилось ранее), для локальной сети и для шлюза по умолчанию. Например, если шлюз по умолчанию имеет адрес 128.17.75.98, можно использовать команду:

```
route add default gw 128.17.75.98
```

`route` имеет несколько параметров. Задание *-net* или *-host* перед адресом назначения (*destination*) указывает `route`, что адрес является адресом сети или отдельного хоста, соответственно. (В большинстве случаев определяются маршруты к сетям, но иногда может понадобиться указать маршрут к отдельной машине. Для такой записи в таблице используется ключ *-host*.)

Параметр *metric* устанавливает значение метрики данного маршрута. Значения метрики используются в тех случаях, когда есть несколько маршрутов к месту назначения и система должна принять решение, какой из них выбрать. Как правило, выбирается маршрут с наименьшим значением метрики. В нашем примере установим значение метрики для маршрута по умолчанию равным 1, и ему будет отдаваться предпочтение перед другими маршрутами.

Каким образом могут появиться несколько маршрутов к одному определенному адресу назначения? Прежде всего, можно использовать несколько команд `route` в *rc.inet1* для одного адреса, например, когда есть несколько шлюзов в некоторую сеть. Однако таблицы маршрутизации могут динамически пополняться новыми записями, если работает `routed` (описываемый ниже). Во время своей работы демон `routed` может создавать новые записи в таблице маршрутизации на вашей машине, если другие системы широковещательно распространяют информацию о маршрутах. Установив значение метрики в 1, вы гарантируете, что новые записи в таблице маршрутизации не получат приоритета над вашим шлюзом по умолчанию.

Вам стоит прочитать страницы справочного руководства команд *ifconfig* и *route*, где вы найдете подробное описание их синтаксиса. Возможно, в вашем случае подойдут другие параметры этих команд.

Пойдем дальше. Сценарий *rc.inet2* применяется для запуска различных демонов, используемых TCP/IP. Они не обязательны для работы с сетью и поэтому вынесены в отдельный *rc*-файл. Как правило, необходимо сначала настроить *rc.inet1* и убедиться, что система способна принимать и посылать пакеты в сеть, и лишь затем приступать к настройкам в файле *rc.inet2*.

Сценарий *rc.inet2* запускает следующие демоны: *inetd*, *syslogd* и *routed*. Файл *rc.inet2* может запускать и другие серверы, но мы рекомендуем закомментировать их на время отладки сети.

Наиболее важным среди этих серверов является `inetd`, который выступает в роли оператора для других системных демонов.[1] Он работает в фоновом режиме и прослушивает определенные сетевые порты, ожидая входящие соединения. Когда поступает запрос на соединение, `inetd` запускает новую копию соответствующего демона для этого порта. Например, когда поступает запрос на соединение с сервером FTP, `inetd` запускает `in.ftpd`, который затем управляет FTP-соединением. Это проще и эффективней, чем запускать множество отдельных копий каждого демона. В этом случае сетевые демоны запускаются только при необходимости.

`syslogd` – это демон регистрации системных событий. Он собирает сообщения от различных приложений и записывает их в файл журнала, основываясь на информации о настройках в файле */etc/syslogd.conf*.

`routed` – это сервер, используемый для управления динамической информацией о маршрутизации. Когда система пытается послать пакет в другую сеть, это может потребовать дополнительных записей в таблице маршрутизации. `routed` заботится об управлении таблицей маршрутизации без необходимости вмешательства пользователя.

Ниже приводится пример содержимого файла *rc.inet2*, который запускает демоны `syslogd`, `inetd` и `routed`:

```
#! /bin/sh
# Пример /etc/init.d/rc.inet2

# Запуск syslogd
if [ -f /usr/sbin/syslogd ]
then
/usr/sbin/syslogd
fi

# Запуск inetd
if [ -f /usr/sbin/inetd ]
then
/usr/sbin/inetd
fi

# Запуск routed
if [ -f /usr/sbin/routed ]
then
/usr/sbin/routed -q
fi
```

`named` является одним из многих дополнительных серверов, которые вы, возможно, захотите запустить из сценария *rc.inet2*. *named* – это сервер имен, который отвечает за преобразование IP-адресов (локальных) в имена и наоборот. Использование *named* может понадобиться, если в вашей сети нет сервера имен или вы сами хотите снабжать локальными именами хостов другие машины вашего домена. Настройка *named* достаточно сложна и требует предварительного планирования. Заинтересованных читателей мы отсылаем к книге «DNS and BIND».

---

[1] Почему его и называют часто «суперсервер `inetd`». – *Примеч. науч. ред.*

/etc/hosts

/etc/hosts содержит перечень IP-адресов и имен хостов, которым они соответствуют. Обычно */etc/hosts* содержит только записи для вашей локальной машины и, возможно, других «важных» машин (таких как сервер имен или шлюз). Преобразование адресов в имена для других машин в сети выполняет локальный сервер имен. На самом деле, чтобы обеспечить разрешение сетевых имен в IP-адреса, совсем не обязательно запускать локальный сервер имен, для этих целей достаточно будет одного файла */etc/hosts*. Если в сети имеется всего несколько машин, такой способ определения IP-адресов ничуть не хуже любых других. Но если в сеть объединено достаточно большое число узлов, тогда пришлось бы изменять файл */etc/hosts* на каждой машине при любых изменениях в конфигурации сети, а в этом случае сложность задачи возрастает в квадратичной прогрессии и очень быстро переходит в разряд неосуществимых.

Например, если ваша машина называется *eggplant.veggie.com* и ее IP-адрес – 128.17.75.20, то */etc/hosts* может выглядеть так:

```
127.0.0.1 localhost
128.17.75.20 eggplant.veggie.com eggplant
```

Если вы используете только петлевой интерфейс, единственной строкой в файле */etc/hosts* будет адрес 127.0.0.1.

/etc/networks

Файл */etc/networks* содержит имена и адреса вашей собственной и других сетей. Он используется командой `route` и позволяет указывать сеть по ее имени вместо адреса.

Любая сеть, маршрут для которой вы хотите добавить при помощи команды `route` (обычно вызываемой из *rc.inet1*), должна для удобства иметь запись в */etc/networks*. В противном случае вместо имени вам придется указывать IP-адрес сети.

Например:

```
default 0.0.0.0         # маршрут по умолчанию - обязательно
loopnet 127.0.0.0       # петлевая сеть - обязательно
veggie-net 128.17.75.0  # Изменить на собственный адрес сети
```

Теперь вместо команды

```
route add 128.17.75.20
```

можно использовать

```
route add veggie-net
```

/etc/host.conf

Файл */etc/hosts.conf* указывает, как ваша система разрешает имена хостов. В нем должно быть две строки:

```
order hosts,bind
multi on
```

Эти строки указывают библиотекам распознавания при поиске любых имен сначала проверить файл */etc/hosts*, а затем (если адрес не найден) запросить сервер

имен (в случае наличия такового). Запись `multi` позволяет указать несколько IP-адресов для одного имени хоста в */etc/hosts*.

В системах, использующих новую библиотеку *glibc2* (что относится к большинству новых дистрибутивов), вместо */etc/host.conf* используется */etc/nsswitch.conf*. В этом случае файл должен содержать строки: `hosts: files dns` и `networks: files dns`.

/etc/resolv.conf

Файл */etc/resolv.conf* определяет параметры настройки механизма распознавания сетевых имен, указывая адрес вашего сервера имен (если таковой есть) и домены, которые подставляются по умолчанию, если указанное имя не является полным доменным именем. Например, если этот файл содержит строку

```
search vpizza.com vpasta.com
```

то при использовании имени хоста *blurb* будет сделана попытка найти имена *blurb.vpizza.com* и *blurb.vpasta.com* (в этом порядке). Это довольно удобно, так как не надо набирать полные имена часто используемых доменов. С другой стороны, чем больше доменов вы здесь укажете, тем дольше будет выполняться просмотр DNS.

Например, машина *eggplant.veggie.com* с сервером имен по адресу 128.17.75.55 будет иметь следующие строки в файле */etc/resolv.conf*:

```
domain      veggie.com
nameserver  128.17.75.55
```

Можно указать более одного сервера имен, для каждого из которых должна быть задана строка *nameserver* в *resolv.conf*.

Установка имени хоста

Необходимо установить имя хоста для вашей системы (hostname) с помощью команды `hostname`. Обычно она вызывается из */etc/init.d/rc.sysinit*. Чтобы выяснить, откуда она вызывается, просто просмотрите свои системные *rc*-файлы. Например, если имя вашего хоста (полное) – *eggplant.veggie.com*, отредактируйте соответствующий *rc*-файл, чтобы он выполнял команду */bin/hostname eggplant.veggie.com*. Обратите внимание: исполняемый файл `hostname` в вашей системе может оказаться не в */bin*, а в другом каталоге.

Проверяем сеть

После того как вы изменили все это множество файлов конфигурации, вы должны быть готовы перезагрузить систему (используя ядро с поддержкой TCP/IP) и попытаться использовать сеть.

При первой загрузке системы вы можете запретить выполнение *rc.inet1* и *rc.inet2* и запустить их вручную после загрузки. Это позволит отслеживать сообщения об ошибках, изменять сценарии и делать повторные попытки. Когда вы будете удовлетворены устойчивой работой этих настроек, можно разрешить выполнение сценариев из */etc/inittab*.

Хорошим способом проверки сетевого соединения является простое `ssh`-подключение к другому хосту. Сначала попытайтесь подключиться к хосту в вашей локальной сети, и если это удастся, попробуйте соединения с хостами из других се-

тей. В первом случае проверяется соединение с локальной подсетью, а во втором – с остальным миром через ваш маршрутизатор.

Может оказаться, что вы подключаетесь к удаленным хостам через шлюз, в то время как соединиться с локальными машинами не удается. Это означает ошибку в маске подсети или записи для локальной сети в таблице маршрутизации.

При проверке соединения с другими машинами сначала попытайтесь использовать только IP-адрес удаленного хоста. Если это удалось, а соединение по имени хоста не устанавливается, возможно, что-то не так с настройками сервера имен (например, */etc/resolv.conf* и */etc/host.conf*) или с маршрутом к серверу имен.

Самым частым источником проблем с сетью является неправильно настроенная таблица маршрутизации. Просмотреть таблицу маршрутизации можно с помощью команды:

```
netstat -rn
```

В предыдущем разделе мы рассмотрели формат таблицы маршрутизации, которая выводится этой командой. Просмотр страниц справочного руководства для *netstat*(8) будет также полезным. Команда *netstat* без параметра *-n* показывает имена хостов и сетей вместо их IP-адресов.

Для отладки таблиц маршрутизации можно либо редактировать *rc.inet1* и затем перегружаться, либо вручную добавлять и удалять записи, используя команду `route`. Полный синтаксис команды `route` вы найдете на странице справочного руководства `route`(8). Обратите внимание, что просто отредактировав и перезапустив *rc.inet1*, вы не сотрете старые записи в таблице маршрутизации; необходимо либо перезагрузить систему, либо использовать команду `route del` для удаления этих записей.

Если вам кажется, что совсем ничего не работает, возможно, есть проблемы с настройкой вашего устройства Ethernet. Для начала проверьте, была ли обнаружена ваша сетевая карта по соответствующему адресу и (или) IRQ во время загрузки. Эту информацию можно получить из сообщений при загрузке ядра, а если вы используете *syslogd*, загрузочные сообщения ядра сохраняются также в файле, например в */var/log/messages*.

Хороший способ определить, действительно ли проблемы вызваны картой Ethernet, – воспользоваться командой `ifconfig` *`имя_интерфейса`*, например:

```
ifconfig eth0
```

В результате будут показаны статистические сведения об интерфейсе. Если видно, что интерфейс принимал и посылал пакеты, значит, ядро его видит и существенной проблемы с аппаратурой нет. Если после выполнения команды

```
ifconfig
```

вы не видите своей карты в списке, значит, ядро даже не увидело ее.

Если сетевая карта обнаруживается неправильно, может потребоваться изменить параметры ядра и исправить этот недостаток. Много информации по настройке сетевых карт содержится в «Linux Ethernet HOWTO». В большинстве случаев исправить ошибку просто. Для этого достаточно установить соответствующее прерывание и адрес порта в загрузочной строке LILO. Например, при загрузке через LILO с командой:

```
lilo: linux ether=9,0x300,0,1,eth0
```

будут выбраны IRQ 9, базовый адрес 0x300 и внешний трансивер (четвертое значение, установленное в 1) для устройства *eth0*. Если вы используете внутренний трансивер (когда карта поддерживает оба типа), измените четвертое значение параметра `ether` на 0.

Кроме того, не стоит недооценивать возможность неисправности сетевой карты или неправильного ее подключения к машине или к сети. Неисправность сетевой карты или кабеля может быть причиной бесконечных неприятностей, включая неустойчивую работу сети, отказы системы и т. д. Когда вы перепробовали уже все, замените сетевую карту и (или) кабель и выясните, не это ли было источником проблемы.[1]

Если ваша сетевая карта распознается, но в системе по-прежнему имеются проблемы, связанные с сетью, причиной может быть неверная настройка устройства с помощью *ifconfig*. Убедитесь, что вы правильно указали IP-адрес, широковещательный адрес и маску подсети. Вызов `ifconfig` без аргументов выдает информацию о настройках устройства Ethernet.

Коммутируемые линии и протокол PPP

Для связи по TCP/IP с помощью модема (например, через подключение по коммутируемому каналу к провайдеру Интернета) или с помощью другого последовательного устройства (такого, как «нуль-модемный» кабель между двумя машинами) Linux имеет программную поддержку протокола PPP. PPP – это протокол, который преобразует пакеты, передаваемые по сети (например, TCP/IP), в формат, легко передаваемый через модем или последовательный кабель. Скорее всего, сервер вашего провайдера Интернета использует PPP для обслуживания подключений по коммутируемым линиям. Настроив PPP под Linux, вы сможете напрямую соединиться с провайдером.

SLIP – это более старый протокол, в котором есть все основные функции, выполняемые PPP. Однако ему не хватает некоторых важных возможностей, таких как управление IP-адресами и размерами пакета. В наши дни SLIP практически полностью вытеснен протоколом PPP.

В этом разделе мы рассмотрим настройку *PPP-клиента*, то есть системы, которая подключается к провайдеру Интернета (или к другому PPP-серверу) для того, чтобы выйти во внешний мир. Можно настроить Linux-машину в качестве PPP-сервера, но это более сложная задача, о которой рассказывается в «Linux Network Administrator's Guide».

Основные настройки PPP для модемов

В большинстве частей света люди используют обычные модемы с дозвоном для передачи цифровых данных по коммутируемым телефонным линиям. Поэтому в первую очередь мы рассмотрим настройку модемов. Затем покажем, как на-

[1] Однажды один из авторов три часа подряд пытался выяснить, почему ядро не распознает сетевую карту. Как оказалось, 16-разрядная карта была вставлена в 8-разрядный разъем – *сам виноват*.

строить PPP для более быстрого и удобного типа линий, называемого ISDN (Integrated Services Digital Network – цифровая сеть с предоставлением комплексных услуг) и весьма популярного в Европе, а также имеющегося в США, но недостаточно там рекламируемого.

Требования

Большинство систем Linux поставляются с предустановленным программным обеспечением, необходимым для работы с PPP. Главное, что вам нужно, – это ядро, скомпилированное с поддержкой PPP, демон *pppd* и различные средства, имеющие отношение к PPP, включая программу `chat`.

В большинстве дистрибутивов Linux поддержка PPP включена в предварительно скомпилированное ядро или в модуль ядра, подключаемый по требованию. Однако может оказаться необходимым самостоятельно скомпилировать поддержку PPP. Для этого требуется лишь включить относящиеся к PPP параметры настройки ядра и пересобрать его. Обычно поддержка PPP компилируется в виде отдельного модуля, поэтому в этом случае достаточно скомпилировать только модули ядра. Информацию по сборке ядра и модулей вы найдете в разделе «Сборка нового ядра» главы 18.

Утилиты `pppd` и `chat` являются приложениями пользовательского уровня, контролирующими использование PPP на вашей системе. Они входят практически во все дистрибутивы Linux. В системах Red Hat эти утилиты устанавливаются в */usr/sbin* и находятся в RPM-пакете `ppp`.

Кроме того, для использования PPP необходим модем, совместимый с Linux и типом модемов, используемых на сервере вашего провайдера. В эту категорию попадает большинство стандартных модемов 14,4, 28,8, 56K и другие. Есть очень немного типов модемов, не поддерживаемых Linux, а что касается провайдеров, то они обычно не используют ничего настолько эзотерического, что заставило бы вас купить особый модем.

Единственный тип модемов, которого надо остерегаться, – это так называемый Winmodem. Первоначально этот модем продавался компанией US Robotics, но теперь его выпускают и другие производители. Winmodem использует центральный процессор компьютера для преобразования цифровых сигналов в аналоговые, чтобы их можно было посылать по обычным телефонным линиям, в отличие от обычных модемов, где есть специальный чип, который выполняет эту функцию. Проблема с Winmodem заключается в том, что информация о программировании этих устройств закрыта их изготовителями, а потому написание драйверов под Linux для этого класса устройств затруднительно. (Кроме того, многие издеваются над идеей тратить драгоценные циклы CPU на производство модемных сигналов, то есть на работу, которую лучше оставить специализированному оборудованию. С другой стороны, обновление так называемых программных модемов состоит в замене драйвера операционной системы, который управляет их работой. Это существенное преимущество по сравнению с альтернативой покупки нового оборудования.)

Имена последовательных устройств

Под Windows 95/98/ME и MS-DOS модемы и другие последовательные устройства называются COM1 (для первого устройства), COM2 (для второго) и так далее

до COM4. (Большинство систем поддерживают до четырех последовательных устройств, но существуют многопортовые карты, которые могут увеличить это число.) Под Linux эти же устройства называются */dev/ttyS0*, */dev/ttyS1* и так далее до */dev/ttyS3*.[1] В большинстве систем во время установки создается символическая ссылка */dev/modem*, которая указывает на последовательное устройство, где находится модем, как это показано в следующем листинге:

```
% ls -l /dev/modem
lrwxrwxrwx 1 root root 10 May 4 12:41 /dev/modem -> /dev/ttyS0
```

Если для вашей системы эта ссылка не подходит (допустим, вы знаете, что ваш модем связан не с устройством */dev/ttyS0*, а с */dev/ ttyS2*), вы легко можете изменить ее, зарегистрировавшись как *root* и введя

```
# ln -sf /dev/ttyS2 /dev/modem
```

Установка PPP

Настройка PPP состоит из нескольких шагов. Возможно, в первую очередь следует проверить наличие в дистрибутиве некоторого рода мастеров, с помощью которых можно выполнить настройку PPP. С другой стороны, выполнив все необходимые настройки вручную, можно будет узнать о принципах работы с PPP гораздо больше. Первое, что необходимо сделать при выполнении настроек вручную, – это написать сценарий chat, который выполняет начальный обмен информацией (так называемое «рукопожатие» – handshaking), необходимый для установки PPP-соединения между вашей машиной и провайдером. В процессе «рукопожатия» передается различная информация, такая как регистрационное имя и пароль. Второй шаг – это написание сценария, который запускает демон `pppd`. Этот сценарий заставляет модем звонить провайдеру и запускает PPP. На последнем шаге редактируется системный файл */etc/resolv.conf* для того, чтобы машина знала, где находится сервер доменных имен (DNS). Мы последовательно рассмотрим эти шаги.

Прежде всего, необходимо узнать следующую информацию:

- Номер дозвона до провайдера интернет-услуг (ISP).
- Регистрационное имя и пароль для связи с провайдером.
- IP-адрес DNS вашего провайдера.

Эта информация предоставляется провайдером на этапе создания учетной записи. Кроме того, может понадобиться следующая информация:

- IP-адрес сервера провайдера.
- IP-адрес вашей системы (если он не выделяется провайдером динамически).
- Маска подсети, которую вам следует использовать.

Последние три пункта обычно автоматически определяются при установке PPP-соединения, однако иногда этого не происходит. Так что вам не повредит иметь эти данные на случай, если они понадобятся.

---

[1] Старые версии Linux использовали также специальные устройства «callout», к которым обращались по именам от */dev/cua0* до */dev/cua3*. Начиная с ядра версии 2.2 они не используются.

Создание сценария для chat

chat – это программа, которая осуществляет обмен некоторой информацией (например, передачу регистрационного имени и пароля) между клиентом и сервером PPP при установке соединения. Кроме того, `chat` отвечает за набор номера провайдера модемом и некоторые другие простые задачи.

Программа *chat* автоматически вызывается *pppd* при запуске (мы обсудим это позже). Вам необходимо лишь написать простой сценарий оболочки, который вызывает `chat` для управления подключением. Следующий пример является простым сценарием *chat*. Создайте (обладая привилегиями пользователя *root*) файл */etc/ppp/my-chat-script* и введите туда следующие строки:

```
#!/bin/sh
# my-chat-script: программа дозвона до провайдера
exec chat -v    \
'' ATZ          \
OK ATDT555-1212 \
CONNECT ''      \
ogin: mdw       \
assword: my-password
```

Задание `ogin` и `assword` (без начальных букв) позволяет ждать приглашения, начинающиеся как с прописных, так и строчных букв.

Убедитесь, что файл *my-chat-script* является выполняемым. Сделать его выполняемым можно командой *chmod 755 /etc/ppp/my-chat-script*.

Обратите внимание на то, что если строка заканчивается символом обратного слэша, то *после* него не должно быть никаких символов. Обратный слэш в сценариях оболочки означает продолжение на следующей строке.

Третья строка сценария запускает саму программу `chat` с параметрами, указанными в следующих строках. Каждая строка содержит два поля, разделенные пробелами, – «ожидаемая» строка и «посылаемая» строка. Идея состоит в том, что сценарий `chat` должен отвечать «посылаемой» строкой на получаемую от модемного соединения «ожидаемую» строку. Например, последняя строка сценария информирует `chat` о том, что следует ответить строкой `my-password` при запросе `assword`, полученном от сервера провайдера.

Первая строка сценария дает указание программе `chat` послать модему команду `ATZ`, что должно вызвать повторную инициализацию модема. (В «ожидаемой» строке стоит `''`. Это означает, что до команды `ATZ` ничего не ожидается.) Вторая строка ожидает от модема ответ `OK`, после чего набирается номер с использованием строки `ATDT555-1212`. (Если ваша АТС использует импульсный набор номера, измените эту строку на `ATDP555-1212`. Номер телефона, конечно, нужно указать тот, где находится модемный пул провайдера.)

Когда модем отвечает сигналом `CONNECT`, посылается перевод строки (указанный как `''`). После этого `chat` ожидает запрос `ogin:`, прежде чем послать регистрационное имя, а затем ожидает строку `assword:` перед отправкой пароля.

Строки рассматриваемого примера, начинающиеся с `AT`, являются строками управления (`AT`-командами) для стандартных модемов типа Hayes. В руководстве к вашему модему должно объясняться их использование; оно не зависит от операционной системы (Linux или другой). Например, запятая, поставленная

в набираемом номере, заставит модем сделать паузу перед набором остальных цифр. Если для доступа к внешней линии надо набрать специальную цифру (в данном примере 9), можно послать модему строку `ATDT9,,,555-1212`.

Обратите внимание, что это очень простой сценарий `chat`, не обрабатывающий тайм-ауты, ошибки или другие неординарные ситуации, которые могут возникнуть при попытке дозвониться до провайдера. Поинтересуйтесь на страницах справочного руководства для `chat`, как учесть такие ситуации. Также нужно заранее выяснить, какие строки запросов использует ваш провайдер (мы предполагаем, что это `login` и `password`). Существует несколько способов выяснить это. Возможно, провайдер заранее даст вам эту информацию либо предоставит сценарий для входа для другой системы, такой как Windows 95 (которая использует механизм, очень схожий с `chat`). В противном случае вы можете вручную дозвониться до сервера провайдера, используя простой эмулятор терминала, такой как *minicom* или *seyon*. Страницы справочного руководства для этих команд помогут вам.

Запуск демона pppd

Теперь все готово к тому, чтобы приступить к настройке демона *pppd* для инициирования PPP-соединения с помощью сценария `chat`, который только что написали. Обычно для этого пишут еще один сценарий оболочки, который вызывает *pppd* с набором параметров.

Команда *pppd* имеет следующий формат:

```
pppd device-name baudrate options
```

В табл. 13.1 представлены параметры, поддерживаемые *pppd*. Некоторые из них, скорее всего, вам никогда не понадобятся.

Таблица 13.1. Основные параметры pppd

Параметр	Эффект
`lock`	Блокировать последовательное устройство, запрещая доступ к *pppd*
`crtscts`	Использовать аппаратный контроль потока
`noipdefault`	Не пытаться выяснить локальный IP-адрес. IP-адрес назначается удаленной системой
`user` *username*	Указывает имя хоста или имя пользователя для идентификации по методу PAP или CHAP
`netmask` *mask*	Указывает маску подсети для подключения
`defaultroute`	Добавляет маршрут по умолчанию в таблицу маршрутизации локальной системы, используя IP-адрес удаленной системы в качестве шлюза
`connect` *command*	Использует заданную команду *command* для инициализации соединения. *pppd* предполагает, что этот сценарий находится в */etc/ppp*. Если это не так, следует указать полный путь к сценарию
local_IP_address:remote_IP_address	Указывает локальный и (или) удаленный IP-адрес. Один или оба могут быть указаны как 0.0.0.0, тогда адрес будет назначен удаленной системой
`debug`	Протоколирует информацию о соединении через демон *syslog*

Команда *pppd* обычно вызывается из сценария на языке командной оболочки. Отредактируйте файл */etc/ppp/ppp-on*, добавив в него следующие строки:

```
#!/bin/sh
          # the ppp-on script

          exec /usr/sbin/pppd /dev/modem 38400 lock crtscts noipdefault \
          defaultroute 0.0.0.0:0.0.0.0 connect my-chat-script
```

Как и в предыдущем примере с файлом *my-chat-script*, следует убедиться, что файл является исполняемым, и остерегаться символов после символа обратного слэша в конце строки.

Когда этот сценарий будет готов, вы сможете подключиться к провайдеру, используя команду

```
% /etc/ppp/ppp-on
```

Для выполнения этой команды не нужны привилегии суперпользователя. При запуске сценария вы должны услышать, как модем набирает номер, и если все в порядке, через минуту PPP-соединение будет успешно установлено. Если PPP работает, команда *ifconfig* должна выдать запись для сетевого интерфейса `ppp0`:

```
# ifconfig
lo        Link encap:Local Loopback
          inet addr:127.0.0.1 Bcast:127.255.255.255 Mask:255.0.0.0
          UP BROADCAST LOOPBACK RUNNING MTU:3584 Metric:1
          RX packets:0 errors:0 dropped:0 overruns:0 frame:0
          TX packets:0 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0

ppp0      Link encap:Point-to-Point Protocol
          inet addr:207.25.97.248 P-t-P:207.25.97.154 Mask:255.255.255.0
          UP POINTOPOINT RUNNING MTU:1500 Metric:1
          RX packets:1862 errors:0 dropped:0 overruns:0 frame:0
          TX packets:1288 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0
          Memory:73038-73c04
```

Мы видим, что PPP работает, локальный IP-адрес – 207.25.97.248, а IP-адрес удаленного сервера – 207.25.97.154.

При желании иметь уведомление об установлении PPP-соединения (сценарий *ppp-on* возвращается немедленно) добавьте следующую строку в */etc/ppp/ip-up*:

```
/usr/bin/wall "PPP is up!"
```

/etc/ppp/ip-up выполняется, когда PPP устанавливает IP-соединение, поэтому можно использовать этот сценарий для вызова команды `wall` после подключения.

Другой несложный сценарий оболочки можно использовать для завершения PPP-сессии. Отредактируйте файл */etc/ppp/ppp-off* следующим образом:

```
#!/bin/sh
# Простой сценарий ppp-off

kill `cat /var/run/ppp0.pid`
```

Запуск */etc/ppp/ppp-off* завершит работу демона PPP и закроет модемное соединение.

Настройка DNS

Само по себе использование `pppd` вместе с `chat` лишь устанавливает PPP-соединение и назначает вам IP-адрес. Для использования доменных имен нужно сообщить системе IP-адрес сервера имен, установленного у вашего провайдера. Это можно сделать, отредактировав файл */etc/resolv.conf*, который подробно описан на страницах справочного руководства для *resolver*. Однако для большинства задач достаточно прописать строки двух типов. Один из них служит для указания списка доменов, в котором ведется поиск всякий раз, когда используется доменное имя, а другой тип строк содержит адреса серверов DNS.

Пример файла */etc/resolv.conf* может выглядеть так:

```
# Пример /etc/resolv.conf
search cs.nowhere.edu nowhere.edu
nameserver 207.25.97.8
nameserver 204.148.41.1
```

Первая строка указывает, что каждый раз при использовании доменного имени (такого как *orange* или *papaya*) его следует искать в списке указанных доменов. В этом случае программа, отвечающая за распознавание имен хостов, будет сначала расширять имена типа *papaya* до *papaya.cs.nowhere.edu* и пытаться найти систему по этому имени, а затем при необходимости пробовать вариант *papaya.nowhere.edu* и пытаться найти систему снова.

Строки, начинающиеся с `nameserver`, указывают IP-адреса серверов DNS (они должны быть предоставлены провайдером), которые используются вашей системой для распознавания доменных имен. Если вы укажете более одной строки `nameserver`, указанные серверы будут вызываться по очереди до тех пор, пока один из них не выдаст результат. В этом случае один DNS-сервер считается первичным, а остальные запасными.

Ошибки при настройке PPP

Настройка PPP, описанная здесь, является очень простой и, скорее всего, не подойдет на все случаи жизни. Лучшим источником дополнительной информации являются страницы справочного руководства для `pppd` и `chat`, а также «Linux PPP HOWTO» и относящиеся к ним документы.

К счастью, и *chat*, и *pppd* протоколируют свою работу, а также все возникающие ошибки, используя стандартные возможности демона *syslog*. Отредактировав файл */etc/syslog.conf*, вы можете указать файл, куда эти программы будут записывать сообщения. Для этого добавьте следующие строки:

```
# Сохранять сообщения от chat
local2.* /var/log/chat-log

# Сохранять сообщения от pppd
daemon.* /var/log/pppd-log
```

Теперь сообщения от `chat` будут протоколироваться в файле */var/log/chat-log*, а сообщения от `pppd` – в */var/log/pppd-log*.

Обратите внимание: эти записи будут содержать конфиденциальную информацию, такую как используемые при подключении к провайдеру регистрационные

имена и пароли! Журналирование таких сообщений следует разрешать только на время настройки и отладки PPP. Когда все будет работать, удалите оба файла, а также удалите введенные строки из */etc/syslog.conf*.

`chat` сохраняет некоторые сообщения об ошибках и в файле */etc/ppp/connect-errors*, который не управляется демоном *syslog*. (Этот файл можно спокойно оставить.)

Протоколы PAP и CHAP

Некоторые провайдеры требуют использовать специальные протоколы аутентификации, такие как PAP (Password Authentication Protocol) или CHAP (Challenge Handshake Authentication Protocol). Эти протоколы основаны на некоторых «общих секретах», известных и клиенту, и серверу. В большинстве случаев это просто пароль на доступ к провайдеру.

Если провайдер требует использовать PAP или CHAP, то настроить эти протоколы можно, добавив некоторые данные в файлы */etc/ppp/pap-secrets* или */etc/ppp/chap-secrets*, соответственно. В каждом файле есть четыре поля, разделенных пробелами или табуляцией. Ниже приводится пример файла *pap-secrets*:

```
# Secrets for authentication using PAP
# client    server    secret        IP or Domain
mdw         *         my-password
```

Первое поле содержит имя вашей системы, ожидаемое удаленным сервером. Обычно это ваше регистрационное имя для доступа к провайдеру. Второе поле указывает имя сервера провайдера; звездочка `*` позволяет использовать эту запись для всех серверов, с которыми вы хотите связаться. Третье поле содержит «общий секрет», предоставленный провайдером. Как указывалось ранее, обычно это ваш пароль. Четвертое поле используется провайдерами, чтобы ограничить IP-адреса или имена доменов, к которым могут иметь доступ подключающиеся по модему пользователи. Но для большинства конфигураций PPP-клиентов это поле не требуется.

Файл *chap-secrets* содержит те же четыре поля, но в нем нельзя использовать символ `*` вместо имени сервера провайдера. Это секрет, который провайдер передает вам при открытии у него учетной записи.

Если используются PAP или CHAP, не обязательно включать в сценарий `chat` команды для обмена паролями и регистрационными именами после получения сообщения `CONNECT` — обо всем дальнейшем позаботится *pppd*. Поэтому вы можете отредактировать */etc/ppp/my-chat-script* так, чтобы в нем были только строки:

```
#!/bin/sh
# my-chat-script: программа дозвона до провайдера
exec chat -v    \
'' ATZ          \
OK ATDT555-1212 \
CONNECT ''
```

Кроме того, нужно добавить параметр `user` в строку вызова *pppd* в */etc/ ppp/ppp-on*:

```
#!/bin/sh
# the ppp-on script
exec /usr/sbin/pppd /dev/modem 38400 lock crtscts noipdefault \
user mdw defaultroute 0.0.0.0:0.0.0.0 connect my-chat-script
```

PPP поверх ISDN

Уже многие годы ISDN предлагает удобную, высокоскоростную цифровую связь; она особенно популярна в Европе, где цены и маркетинговая политика сделали ее более привлекательной, чем в США. ISDN, где в одной линии интегрированы передача данных и голоса, предлагает более быструю установку соединения и значительно более высокие скорости передачи данных по сравнению с традиционными модемами.

Линии ISDN могут передавать данные со скоростью 64 Кбит/с. В отличие от аналоговых линий, эта скорость постоянна, так как не зависит от капризов аналоговой передачи, подверженной влиянию разнообразных помех. Более новый протокол, названный ADSL (Asynchronous Digital Subscriber Line), поднимает планку скорости доступа по телефонным линиям еще выше, но ISDN по-прежнему сохраняет большую долю рынка.

В этом разделе мы опишем, как настроить доступ к провайдеру по линии ISDN. Мы рассмотрим только самый обычный способ соединения – синхронный PPP, а не особый режим, называемый Raw IP по ISDN. Более того, в этом разделе обсуждаются только внутренние платы ISDN, которые настраиваются не так, как соединения по телефонным линиям, рассмотренные в предыдущем разделе. Для установки внешних устройств ISDN, или так называемых ISDN-модемов (термин, конечно, является оксюмороном, так как нет модуляции-демодуляции), можно пользоваться командами, подобными тем, которые были описаны в предыдущем разделе, поскольку такое устройство выглядит для операционной системы как обычный модем, предлагающий некоторые дополнительные команды, быструю установку соединения и высокую скорость.

Возможно, вам потребуется дополнительная информация помимо той, которую мы здесь представили. Все информация, касающаяся ISDN для Linux, есть на сайте *http://www.isdn4linux.de* (хотя домен зарегистрирован в Германии, вся информация там на английском языке).

В некотором смысле установка ISDN-соединения значительно проще установки соединения по аналоговым линиям, так как у цифровых линий просто отсутствуют некоторые проблемы (плохие линии связи, длительное время соединения и т. д.). После набора номера соединение устанавливается в течение миллисекунд. Но это может вызвать другие проблемы. Поскольку соединение устанавливается и закрывается так быстро, неправильно настроенная система будет звонить снова и снова, что может дорого обойтись. Это особенно опасно с внутренними адаптерами ISDN, так как вы не услышите щелчков и прочего шума, как у обычных модемов, и нет никаких индикаторов, показывающих состояние соединения. Однако статус линии ISDN можно проверить с помощью простых программ.

Настройка коммутируемого PPP-соединения по ISDN выполняется в два этапа:

1. Конфигурирование оборудования ISDN.
2. Настройка и запуск демона PPP и модификация таблицы маршрутизации для использования линии ISDN.

Мы расскажем об этих шагах в следующих разделах.

Настройка оборудования ISDN

Сначала надо предоставить ядру доступ к плате ISDN. Как и для любого другого оборудования, для вашей платы потребуется драйвер устройства, настроенный соответствующим образом под вашу плату.

Linux поддерживает большое число плат ISDN. Мы не можем рассказать о каждой из них, но процедура настройки для всех плат весьма похожа. Если ваша плата здесь не рассматривается, документацию к ней можно найти в каталоге *Documentation/isdn* в исходных текстах ядра Linux.

Мы будем рассматривать платы, использующие драйвер *HiSax*. Этот драйвер работает почти со всеми картами, которые построены на чипсете HSCX (а это большинство имеющихся на сегодня в продаже пассивных карт). К этой категории относятся, в частности, USR Sportster internal TA и хорошо известные платы Teles, ELSA и Fritz. Другие платы используют схожие настройки. Linux поддерживает даже некоторые активные карты, включая широко известные AVM B1 и IBM Active 2000 ISDN.

Первым делом нужно включить в ядро поддержку ISDN. Рекомендуется скомпилировать все, что связано с ISDN, в виде модулей, особенно на период экспериментов с настройками. Вам потребуется включить сборку в виде модулей следующих элементов ядра:

- ISDN support.
- Support for synchronous PPP.
- Один из драйверов устройств, соответствующий имеющемуся оборудованию. Если вы используете драйвер HiSax, необходимо указать, какую марку платы ISDN вы приобрели, и какой протокол ISDN хотите использовать. Практически всегда используется протокол EURO/DSS1, но если вы живете в Германии и линия ISDN у вас уже давно, протокол может называться 1TR6, а в США – US NI1. Если у вас есть сомнения, обратитесь в телефонную компанию.

Откомпилируйте и установите модули, как описано в главе 18. Теперь вы готовы к настройке оборудования ISDN. В некоторых дистрибутивах, таких как SUSE, настройка линии ISDN отличается простотой и удобством. Мы же пойдем другим путем, предположив, что ваш дистрибутив не столь дружелюбен к пользователям и в нем отсутствует возможность автоматической настройки или вы просто хотите знать, что происходит за кулисами.

Теперь необходимо загрузить модуль драйвера устройства с помощью команды *modprobe*. Она автоматически загрузит все остальные модули. Все драйверы устройств могут принимать ряд входных параметров. Модуль `hisax` среди прочих принимает следующие параметры:

`id=`*`boardid`*

Устанавливает идентификатор для платы ISDN. Вы можете выбрать любое имя, но оно не должно совпадать с идентификаторами других устройств.

`type=`*`boardtype`*

Указывает точную марку и тип платы. Например, значение 16 для *boardtype* выбирает поддержку для USR Sportster internal TA. Полный список типов плат вы найдете в файле *Documentation/isdn/README.hisax*.

`protocol=`*`protocoltype`*

Выбирает подпротокол ISDN. Возможными значениями для *protocoltype* могут быть: 1 – для старого немецкого протокола 1TR6, 2 – для обычного EDSS1 (так называемого Euro ISDN) и 3 – для выделенных линий.

`irq=`*`irqno`*

Указывает используемое прерывание. Не все платы требуют этот параметр.

`io=`*`addr`*

Указывает используемый адрес I/O. Требуется не всеми платами. Некоторым платам требуются два адреса I/O, в этом случае используются параметры `io0` и `io1`.

Например, следующая команда загрузит драйвер HiSax, который будет использоваться с платой Teles 16.3, протоколом Euro ISDN, адресом I/O, равным 0x280, и прерыванием IRQ, равным 10 (весьма часто встречающийся случай):

```
tigger # modprobe hisax type=3 protocol=2 io=0x280 irq=10
```

За дополнительной информацией по вашему оборудованию обращайтесь к *Documentation/isdn/README.HiSax* или другому эквивалентному файлу, соответствующему имеющейся аппаратуре.

Этот модуль не очень разговорчив. Если команда `modprobe` ничего не выдала, возможно, все прошло успешно. На всякий случай можно проверить системный журнал в */var/log/messages*. Там должно быть несколько строк, начинающихся с `HiSax:` (или с имени используемого драйвера) и заканчивающихся строкой:

```
HiSax: module installed
```

Если модуль не загрузился, причину, скорее всего, удастся отыскать в */var/log/messages*. Чаще всего проблемы возникают из-за неправильно указанного адреса I/O или номера прерывания IRQ либо неправильно выбранного типа карты. Если у вас на этой же машине установлена операционная система Windows, загрузите ее и посмотрите, какие она назначает адреса ввода-вывода и номер IRQ. Иногда причину ошибки настройки драйвера удается выяснить с помощью файлов */proc/ioports* и */proc/interrupts*, где можно увидеть, насколько правильно были назначены адреса портов ввода-вывода и номер IRQ для драйвера HiSax.

Прежде чем переходить к следующему разделу, можно выполнить еще одну проверку – позвонить самому себе. Это возможно, так как благодаря ISDN в вашем распоряжении всегда две телефонные линии. Поэтому одна из них может использоваться для исходящего звонка, а вторая – для входящего.

Чтобы подсистема ISDN сообщала о том, что происходит с телефонными линиями, надо сделать ее более разговорчивой, чем это предусмотрено по умолчанию. Сделать это можно с помощью трех утилит, входящих в пакет *isdn4k-utils*, который вы легко сможете найти на любимом FTP-сервере, посвященном Linux.

В пакете *isdn4k-utils* среди прочего есть три утилиты: *hisaxctrl* предназначена для настройки драйвера устройства, *isdnctrl* – для настройки высокого уровня подсистемы ISDN, и *isdnlog* – очень полезное средство, протоколирующее все, что происходит с вашими линиями ISDN. В то время как *hisactrl* и *isdnctrl* можно использовать без каких-либо настроек, для *isdnlog* требуется небольшой файл

настройки. На данный момент мы удовлетворимся быстрым решением, но когда ваше ISDN-соединение заработает, вы, возможно, захотите настроить *isdnlog* так, чтобы видеть, куда уходят деньги.

А сейчас скопируйте в */etc/isdn/isdn.conf* один из примеров файлов с настройками, расположенных в пакете *isdnlog*. Вам потребуется изменить, по крайней мере, следующие строки:

`COUNTRYCODE=`

Добавьте сюда телефонный код вашей страны, например 1 для Соединенных Штатов и Канады, 44 для Великобритании, 46 для Швеции и т. д.

`AREAPREFIX=`

Если код области в вашей стране имеет фиксированный префикс, поместите его сюда. Для большинства европейских стран префикс равен 0, для Финляндии – 9, а для Соединенных Штатов, Дании и Норвегии он отсутствует.

`AREACODE=`

Введите сюда код области. Если вы указали `AREAPREFIX` в предыдущем пункте, не вводите его сюда. Например, Стокгольм (столица Швеции) имеет код области 08. В этом случае вы вводите 0 в `AREAPREFIX` и 8 в `AREACODE`.

Когда вы это сделаете, выполните следующие команды, чтобы ваша система ISDN стала более разговорчивой:

```
tigger # /sbin/hisaxctrl boardid 1 4
tigger # /sbin/isdnctrl verbose 3
tigger # /sbin/lsdnlog /dev/isdnctrl0 &
```

Если вы используете драйвер, отличный от HiSax, может потребоваться другая команда. Например, для драйвера PCBit в пакете *isdn4k-utils* есть команда `pcbitctl`.

Теперь можно позвонить самому себе. Попробуйте все ваши MSN (multiple subscriber numbers – множественные номера абонента, которые являются номерами вашего телефона ISDN), для того чтобы посмотреть, может ли плата обнаружить их все. В течение или после звонка проверьте */var/log/messages*. Там должны быть строки следующего вида:

```
Mar 16 18:34:22 tigger kernel: isdn_net: call from 4107123455,1,0 -> 123456
Mar 16 18:34:33 tigger kernel: isdn_net: Service-Indicator not 7, ignored
```

Это говорит о том, что ядро обнаружило голосовой звонок (индикатор службы равен 0) на номер MSN 123456 с телефона 123455 и с кодом области (0)4107.

Обратите внимание, как указан набранный номер, так как эта информация понадобится в дальнейшем. В одних сетях номер посылается с кодом области, а в других без него. В любом случае поздравляем вас с тем, что ваше оборудование ISDN теперь настроено правильно.

Настройка синхронного PPP

Настройка демона PPP вновь состоит из нескольких шагов. В Linux плата ISDN рассматривается как сетевой интерфейс, который нужно настраивать особыми командами. Кроме того, необходимо указать регистрационное имя и пароль, вы-

данные провайдером. По окончании настройки запускается демон `ipppd`, который до запроса на установку соединения скрывается в потемках фонового режима.

Сначала настроим сетевой интерфейс. Для этого требуется несколько команд, которые системные администраторы обычно помещают в сценарий, хранящийся, например, в файле */sbin/pppon*. Ниже приводится пример сценария, который можно откорректировать с учетом своих параметров:

```
/sbin/isdnctrl addif ippp0
/sbin/isdnctrl addphone ippp0 out 0123456789
/sbin/isdnctrl dialmax ippp0 2
/sbin/isdnctrl eaz ippp0 123456
/sbin/isdnctrl huptimeout ippp0 100
/sbin/isdnctrl l2_prot ippp0 hdlc
/sbin/isdnctrl l3_prot ippp0 trans
/sbin/isdnctrl encap ippp0 syncppp
/sbin/ifconfig ippp0 1.1.1.1 pointopoint 123.45.67.89 metric 1
```

Давайте рассмотрим каждую команду в отдельности:

`isdnctrl addif ippp0`

Сообщает ядру, что будет использоваться новый ISDN-интерфейс с именем `ippp0`. Всегда используйте имена, начинающиеся с `ippp`.

`isdnctrl addphone ippp0 out 0123456789`

Сообщает интерфейсу ISDN, какой телефонный номер следует использовать. Это номер дозвона до провайдера. Если ранее вы использовали доступ по аналоговой линии, узнайте у провайдера, какой номер надо использовать теперь, так как номер для доступа по ISDN может быть другим.

`isdnctrl dialmax ippp0 2`

Указывает, сколько раз следует набирать номер при неудачном соединении, прежде чем прекратить попытки.

`isdnctrl eaz ippp0 123456`

Указывает один из ваших собственных номеров MSN. Это очень важно, так как без него работа будет невозможна. Если провайдер проверяет доступ по номеру, убедитесь, что вы указали тот MSN, который зарегистрировали у провайдера.

`isdnctrl huptimeout ippp0 100`

Указывает время в секундах (последнее число в этой команде), в течение которого линия может бездействовать, перед тем как ядро закроет соединение. Команда не обязательная, но она может сохранить много денег, если у вас нет неограниченного доступа. Если вы забудете закрыть соединение сами, ядро сделает это за вас.

`isdnctrl l2_prot ippp0 hdlc`

Указывает используемый протокол второго уровня. Возможны следующие значения: `hdlc`, `x75i`, `x75ui` и `x75bui`. Большинство провайдеров используют `hdlc`. Если есть сомнения, спросите у провайдера.

`isdnctrl l3_prot ippp0 trans`

Указывает используемый протокол третьего уровня. (`l` в тексте параметра – это буква L.) В настоящее время доступен только `trans`.

`isdnctrl encap ippp0 syncppp`

Указывает используемую инкапсуляцию. Есть несколько возможных значений, но если вы хотите использовать синхронный PPP (или этого требует провайдер), необходимо указать `syncppp`. Другим, не столь частым значением является `rawip`. Но поскольку он обеспечивает очень слабые возможности аутентификации, лишь немногие провайдеры все еще используют его, несмотря на то, что у него немного лучше производительность благодаря меньшей избыточности информации.

`ifconfig ippp0 1.1.1.1 pointopoint 123.45.67.89 metric 1`

Создает новый сетевой интерфейс. Если IP-адрес не присваивается автоматически (как это обычно делается при телефонных подключениях), вместо 1.1.1.1 необходимо указать ваш IP-адрес. Также надо изменить 123.45.67.89 на IP-адрес сервера провайдера.

При желании можно перевернуть всю настройку, написав сценарий, который закрывает интерфейсы и т. д. Например, в нем будет использоваться команда `isdnctrl delif`. Но строгой необходимости в таком сценарии нет, если только вы не хотите отключить все телефонные соединения на этапе выполнения.

Но это еще не все! Необходимо настроить сам демон `ipppd`, что делается в файле */etc/ ppp/ioptions*. Кроме того, можно создать отдельные файлы конфигурации для каждого демона `ipppd`, но это необходимо только в случае, если вы хотите использовать разные соединения ISDN, то есть если у вас есть несколько учетных записей для доступа по линии.

Ниже приводится пример файла *ioptions*, который является достаточно универсальным для работы с большинством провайдеров. Он не дает максимальной производительности, но зато весьма стабилен. Если вы хотите его оптимизировать, спросите вашего провайдера о возможных параметрах и прочтите страницы справочного руководства для `ipppd`(8):

```
debug
/dev/ippp0
user yourusername
name yourusername
mru 1500
mtu 1500
ipcp-accept-local
ipcp-accept-remote
noipdefault
-vj -vjccomp -ac -pc -bsdcomp
defaultroute
```

Здесь необходимо изменить только две вещи: нужно заменить *yourusername* в третьей и четвертой строках на имя пользователя, предоставленное провайдером для связи с его системой. Мы не будем объяснять значение всех параметров; почитайте страницы справочного руководства в случае сомнений.

Доступ по ISDN требует тех же параметров безопасности, что и модемное подключение. С инструкциями по настройке файла *pap-secrets* или *chap-secrets* можно познакомиться в разделе «Протоколы PAP и CHAP» ранее в этой главе.

Теперь все собрано воедино. Сначала запустим демон `ipppd`:

```
tigger # /sbin/ipppd pidfile /var/run/ipppd.ippp0.pid file /etc/ppp/ioptions &
```

Демон *ipppd* будет теперь ожидать запрос на соединение. Так как мы еще не настроили автоматическое подключение, запустим его вручную с помощью следующей команды:

```
tigger # isdnctrl dial ippp0
```

Теперь следует проверить */var/log/messages*. Там должно быть много сообщений, начинающихся с *ipppd*. Последнее из них должно содержать слова `local IP address` и `remote IP address` вместе с IP-адресами. Если вы нашли эти сообщения, значит, все готово. Так как ранее был использован параметр *defaultroute*, ядро настроено на использование ISDN-соединения как маршрута по умолчанию, и поэтому теперь у вас есть доступ к огромному миру Интернета. Начните с проверки IP-адреса провайдера командой `ping`. При желании закрыть соединение введите:

```
tigger # isdnctrl hangup ippp0
```

Если у вас имеется неограниченный доступ в Интернет, тогда можно будет установить значение 0 для параметра *huptimeout*, и соединение уже не будет разрываться автоматически. Это, конечно, хорошо, но следует помнить, что большинство провайдеров сбрасывают соединение каждые 24 часа, а в этом случае, если вам предоставляется динамический IP-адрес, возможно, что каждые 24 часа системе будет назначаться новый IP-адрес.

А если оно не работает?

Если соединение не работает несмотря на то, что оборудование успешно распознано и вы выполнили все описанные настройки, на помощь снова придет */var/log/messages*. Весьма возможно, что там вы найдете причину ошибки, даже если она будет скрыта под ворохом других сообщений.

Самая обычная ошибка – это неправильно указанное имя пользователя или пароль. Если у вас есть проблемы с аутентификацией, в файле журнала вы найдете строку вида:

```
PAP authentication failed
```

или

```
CHAP authentication failed
```

Тщательно проверьте *chap-secrets* или *pap-secrets*. Провайдер по своим файлам журналов тоже может определить, где именно произошла ошибка аутентификации.

Конечно, возможен вариант, что ваш провайдер не поддерживает синхронный PPP, хотя в наши дни большинство провайдеров это делают. В этом случае за точными настройками обращайтесь к провайдеру.

Если соединение по-прежнему не работает, обратитесь за помощью к провайдеру. У хорошего провайдера должна быть служба поддержки, которая поможет подключить вашу Linux-систему. Если провайдер заявляет, что он поддерживает только Windows, значит, пора сменить провайдера. Сейчас много провайдеров, нормально относящихся к Linux. Часто сам персонал службы поддержки

использует Linux и может помочь вам, даже если официальная политика провайдера не предполагает поддержки Linux.

Если по какой-либо причине приходится работать с недружелюбным провайдером, попробуйте найти других клиентов этого же провайдера, использующих Linux. Настройка соединения в нестандартных случаях – это сложная игра с ключами и параметрами демона `ipppd` и подсистемы ISDN в ядре, и если кто-нибудь уже знает, как это делать, следует этим воспользоваться.

Что дальше?

Когда ISDN-соединение работает и есть доступ в Интернет, можно выполнить некоторые настройки для удобства или внести усовершенствования в работу соединения. Ниже приводятся некоторые предложения:

- Заставьте *ipppd* автоматически набирать номер провайдера. Это можно сделать, установив маршрут по умолчанию на устройство `ippp0`:

  ```
  /sbin/route add default netmask 0.0.0.0 ippp0
  ```

  Теперь, когда ядру потребуется отправить IP-пакет на IP-адрес, для которого нет специального маршрута, он заставит демон *ipppd* установить соединение. Используйте эту возможность только в том случае, если вы установили также параметр *huptimeout* в подсистеме ISDN, иначе придется отдать много денег телефонной компании (если у вас нет неограниченного доступа).

  Такая конфигурация может быть опасной для вашего кошелька, так как есть программы, пытающиеся время от времени установить соединение с Интернетом (Netscape – один из первых кандидатов). Если вы все же используете подобную конфигурацию, чаще проверяйте состояние соединения (подробнее – далее в этом разделе).

- Попробуйте воспользоваться утилитами мониторинга ISDN-соединения. Многие из таких инструментов входят в пакет *isdn4k-utils*, включая средства командной строки `imon` и `imontty`, а также средства с графическим интерфейсом X.

- Настройте *isdnlog* для протоколирования именно того, что вам нужно, и получайте с помощью *isdnrep* подробный отчет об использовании линии ISDN. Отчет включает не только звонки с компьютерной системы и на нее, но и звонки на другие устройства ISDN, такие как телефоны и факсы. Есть только одно предостережение: плата ISDN не может фиксировать исходящие телефонные соединения, сделанные с других устройств. Но этот сервис предоставляют многие телефонные компании, отправляя сигнал обратно для того, чтобы ваша система могла зарегистрировать номер. Этот сервис часто предоставляется бесплатно или за символическую плату. Поинтересуйтесь об этом у вашей телефонной компании.

- Для истинных искателей приключений: эксперимент с Multilink-PPP. Как известно, с ISDN у вас есть как минимум две линии. Если вам нужен канал повышенной емкости, почему бы не использовать обе линии? Это именно то, что делает Multilink-PPP. Для его использования надо включить параметр `Support generic MP` во время настройки ядра, а также посмотреть файлы *Documentation/isdn/README.syncppp* и *Documentation/isdn/syncppp.FAQ* в исходных текстах ядра, чтобы воспользоваться советами, как это лучше сделать. Конечно, эту возможность также должен поддерживать и ваш провайдер.

ADSL

Скорость в 64 Кбит/с, предоставляемая ISDN, это прекрасно, но если вам нужен доступ к мультимедийным файлам или вы просто интенсивно используете Интернет, то может потребоваться еще более высокая пропускная способность. Для этого не обязательно тянуть новые кабели в квартиру или офис – ADSL (Asynchronous Digital Subscriber Line) служит удобной альтернативой, дающей пропускную способность, в 128 раз превышающую ту, которую дает стандартный доступ по коммутируемой линии (эта цифра зависит от вашего провайдера и условий аренды), и достигаемую на обычной телефонной линии. Недостаток ADSL в том, что до ближайшей телефонной станции должно быть не более 5–8 километров (3–5 миль), в зависимости от качества кабеля, проложенного до телефонной станции, поэтому в сельской местности такой сервис недоступен. Типичная пропускная способность по каналу ADSL составляет от 0,5 до 8 Мбит/с (мегабит в секунду) для входящего трафика и от 0,125 до 1 Мбит/с – для исходящего (отправляемого вашим компьютером, в том числе это относится и к исходящим сообщениям электронной почты). Существуют другие технологии, названия которых звучат очень похоже, например SDSL. На физическом уровне они существенно отличаются, но настройка их для работы с Linux-машиной не должна отличаться от ADSL.

ADSL не требует дозвона: подключившись к своей учетной записи, вы соединены постоянно. Некоторые провайдеры периодически обрывают соединение (обычно раз в сутки), и тогда вы снова должны зарегистрироваться, чтобы получить доступ к сети.[1]

Как уже говорилось, не существует каких-либо карт или драйверов ADSL. Что касается оборудования, то ADSL представляет собой обычное соединение Ethernet с теми же кабелями.

Способ подключения Linux-машины к линии ADSL определяется вашим провайдером. Некоторые провайдеры предоставляют все необходимое оборудование, например ADSL-модем и ADSL-маршрутизатор, в аренду. Другие провайдеры требуют, чтобы вы купили оборудование – самостоятельно или у того же провайдера. Всю необходимую информацию предоставит вам сам провайдер.

Есть два способа подключения по ADSL – непосредственно к модему ADSL или к промежуточному маршрутизатору ADSL. Если у вас стоит маршрутизатор ADSL (со встроенным модемом ADSL или без него), вы просто подсоединяете к нему свой кабель Ethernet и можете работать. Если вы хотите подключить Linux-машину непосредственно к модему ADSL, то также соединяете компьютер с модемом при помощи кабеля Ethernet, но вам потребуется запустить на машине специальный протокол – PPPoE (PPP over Ethernet). Этот протокол поддерживается особым демоном с именем *pppoed*. Способ его установки зависит от вашего дистрибутива, где должна быть соответствующая документация. Некоторые дистрибутивы дают возможность установить PPPoE в своей программе настройки.

Наконец, следует отметить, что существуют, хотя их немного, особые модемы ADSL, которые подключаются не через Ethernet, а кабелем USB. Технически эта

[1] Причина такого ограничения заключается в том, чтобы не дать вам возможности запустить в работу свой сервер и принудить к заключению более дорогостоящего «бизнес»-соглашения, если у вас появится такая необходимость.

идея слаба, и таких модемов нужно избегать, но если уж вы связались с одним из них, то дополнительные сведения и драйвер, осуществляющий PPPoE через USB (что следовало бы назвать PPP over Ethernet over USB!), можно найти на *http://eciadsl.flashtux.org and http://speedtouch.sourceforge.net*.

Каким бы способом вы ни подключались к ADSL (с маршрутизатором ADSL или без него), необходимо задать правильный IP-адрес. Он может быть статическим (тогда вы должны найти его в информации, полученной от провайдера) или динамическим, назначаемым через протокол DHCP (Dynamic Host Communication Protocol), и тогда вы можете запросить динамический IP-адрес с помощью команды:

```
dhclient eth0
```

Конечно, если у вашей Ethernet-карты другое имя, нужно заменить им `eth0`. Вместо *dhclient* можно воспользоваться утилитой *pump*. Существует также демон DHCP с именем *dhcpcd*, который выполняется в фоновом режиме и при необходимости назначает динамические IP-адреса.

Наконец, многие провайдеры требуют, чтобы время от времени вы активизировали свою линию. Способ определяется вашим провайдером Интернета. Часто для этого требуется зайти на некоторый веб-сайт и ввести идентификационные данные, предоставленные вам провайдером. Как отмечалось, может потребоваться периодически повторять эту процедуру.

Кабельные модемы[1]

При желании доступ в Интернет можно организовать по любому проводу. В настоящее время ведутся интенсивные эксперименты по использованию для этих целей даже электрической сети. Поэтому совершенно не удивительно, что компании, сделавшие бизнес на доставке видеосигнала с арен спортивных событий (фирмы, занимающиеся кабельным телевидением), поняли, что один из своих каналов они могут выделить для организации локальной сети, несущей цифровые данные, благодаря чему появился кабельный доступ в Интернет.

Пропускная способность кабельной сети теоретически может достигать 10 Мбит/с (как старый добрый Ethernet на коаксиальном кабеле), а некоторым поставщикам удалось достигнуть еще более высокой скорости. Обычно провайдер создает сеть узлов, каждый из которых обладает общей пропускной способностью до 10 Мбит/с. Клиенты, использующие один и тот же узел, делят полосу пропускания между собой. Так, если узлом пользуются два клиента, каждый из них получит по 5 Мбит/с. Кроме того, часть клиентов могут быть подключены к центральному узлу (к серверу, который подключается к Интернету, как правило, через оптоволоконный кабель) и получить в свое распоряжение канал с большей полосой пропускания, чем на периферии. Поэтому, прежде чем заключать договор с компанией на оказание услуг доступа в Интернет, следует внимательно изучить возможности кабельной сети. В первую очередь было бы желательно побеседовать со специалистами из отдела технической поддержки. Если они не смогут ответить на какие-либо вопросы, это достаточно точно будет характеризовать уровень обслуживания, который вы будете получать позже.

---

[1] Имеется в виду связь через сети кабельного телевидения. – *Примеч. науч. ред.*

Провайдеры, предоставляющие клиентам доступ в Интернет через кабельные модемы, обычно берут на себя ответственность за обслуживание кабельной линии вплоть до разъема карты Ethernet. Они выдают кабельный модем, подключенный к кабельной сети коаксиальным кабелем, и разъем Ethernet (RJ45) на стороне клиента. Прежде чем браться за настройку подключения к Интернету, необходимо узнать IP-адрес кабельного модема, маску сети и IP-адрес шлюза. Всю эту информацию должен предоставить провайдер, вместе с кабельным модемом. После этого на стороне клиента останется только запустить в работу карту Ethernet (предварительно выполнив все необходимые настройки в соответствии с информацией, полученной от провайдера):

```
/sbin/ifconfig eth0 IP_address netmask
```

и сообщить ядру информацию о шлюзе:

```
/sbin/route add default gw gateway metric 1
```

Эти команды используются для настройки автономной рабочей станции, работающей под управлением Linux. Если позади нее планируется организовать небольшую локальную сеть, необходимо будет настроить маскирование локальных IP-адресов, как это было описано ранее в данной главе. Всю необходимую для этого информацию можно найти в книгах, охватывающих вопросы настройки брандмауэров в Linux, например «Linux Network Administrator's Guide» и «Linux iptables Pocket Reference» (O'Reilly). В некоторых[1] дистрибутивах Linux, таких как Slackware, функция перенаправления IP-пакетов (forwarding) по умолчанию отключена, то есть маскирование работать не будет. Если дело обстоит именно так, нужно добавить следующую строку в сценарий начального запуска:

```
echo 1 >/proc/sys/net/ipv4/ip_forward
```

Все кабельные модемы допускают возможность удаленной настройки. Если вам не повезет и ваш провайдер не настроит кабельный модем должным образом, тогда вам придется выполнить все необходимые настройки самостоятельно – это потребует более глубокого понимания принципов работы TCP/IP, и возможно, вам придется искать помощи у вашего провайдера (или у того, кто действительно сможет выполнить настройку модема).

В некоторых случаях кабельный модем настраивается таким образом, что взаимодействовать с ним можно будет только посредством определенной карты Ethernet, тогда вам придется передать своему провайдеру MAC-адрес вашей карты. В этом случае, если вы поменяете карту Ethernet (или компьютер со встроенной картой Ethernet), вам придется повторно просить провайдера выполнить настройку модема.

Инструменты диагностики сети

Существует большое число инструментальных средств, с помощью которых можно выполнить диагностику проблем, возникающих в сети. В этом разделе мы

[1] В большинстве дистрибутивов включение перенаправления по умолчанию при неправильных базовых настройках сетевой подсистемы может приводить к очень серьезным проблемам, таким как петлевые маршруты в сети. – *Примеч. науч. ред.*

рассмотрим три таких инструмента, которые могут быть полезны при диагностике различных неполадок с сетью.

ping

Самый первый инструмент, который будет рассмотрен, называется *ping*. Команда *ping* посылает так называемые пакеты ICMP серверу, который будет указан в параметрах команды, сервер возвращает их, а `ping` определяет, какое время потребовалось пакету, чтобы дойти до сервера и вернуться обратно. Таким способом можно проверить качество соединения с Интернетом, но чаще эта команда используется для проверки возможности установить соединение с кем-либо. Чтобы убедиться в наличии соединения с Интернетом, достаточно послать ICMP-пакет какому-нибудь компьютеру в Интернете, например:

```
kalle@tigger:~> ping www.oreilly.com
PING www.oreilly.com (208.201.239.36) 56(84) bytes of data.
64 bytes from www.oreillynet.com (208.201.239.36): icmp_seq=1 ttl=46 time=280 ms
64 bytes from www.oreillynet.com (208.201.239.36): icmp_seq=2 ttl=46 time=250 ms
64 bytes from www.oreillynet.com (208.201.239.36): icmp_seq=3 ttl=46 time=244 ms

--- www.oreilly.com ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 2001ms
rtt min/avg/max/mdev = 244.976/258.624/280.430/15.586 ms
```

Обратите внимание: в данном примере через несколько секунд была нажата комбинация клавиш Ctrl+C – было бы не слишком учтиво использовать чужой сервер с целью проверки. Какие выводы можно сделать из этого примера? Прежде всего, мы выяснили, что действительно обладаем возможностью войти в контакт с компьютером в Интернете. Так как в этом примере вводилось сетевое имя сервера, а не его IP-адрес, мы убедились, что система разрешения имен DNS работает. В первой строке был выведен IP-адрес, который принадлежит *www.oreilly.com*. Каждая следующая строка вывода соответствует одному посланному пакету, откуда можно увидеть, как долго путешествовал пакет туда и обратно. Конечно же, времена, которые показаны здесь, могут существенно отличаться от тех, что можете получить вы, поскольку они зависят от удаленности сервера в сети. Обратите также внимание на значение `icmp_seq`. Каждый пакет получает свой порядковый номер, и все пакеты с теми же порядковыми номерами должны быть получены обратно. Если этого не происходит, если есть промежутки в последовательности пакетов, это означает, что соединение с данным узлом сети неустойчиво, кроме того, это может означать, что сервер перегружен и просто оставляет пакеты без внимания.

Следует отметить, что утилита *ping* – недостаточно надежный инструмент для диагностики проблем, связанных с сетью. Отсутствие ответных пакетов может кроме всего прочего означать, что сервер просто не отвечает на пакеты ICMP – ни один сервер не обязан этого делать, и в некоторых случаях именно так и делается, чтобы снизить нагрузку на сервер и повысить уровень его безопасности (если вы заранее не знаете, что некий сервер точно существует, очень сложно будет причинить ему ущерб). Тем не менее считается хорошим тоном отвечать на запросы, посылаемые утилитой `ping`.

Утилита *ping* также интересна тем, что позволяет увидеть, где именно возникли неполадки. Если *ping* вообще ничего не выводит, выводит `network not reachable`

(сеть недоступна) или другое похожее сообщение, это может означать неправильные настройки вашей системы. В этом случае можно попробовать послать ICMP-пакеты по IP-адресу провайдера, чтобы понять, в каком месте возникает проблема – между локальным компьютером и сервером провайдера или дальше.[1] Если ваша домашняя сеть соединяется с Интернетом с помощью маршрутизатора, можно попробовать послать ICMP-пакеты по его IP-адресу. Если проблемы обнаружатся уже на этом этапе, это может означать неправильные настройки локальной системы или повреждение кабеля. Если маршрутизатор «откликается», но не отвечает сервер провайдера, то причина может крыться в отсутствии соединения с провайдером, например, потому, что неправильно были указаны параметры соединения или сервер провайдера был остановлен, или возникли проблемы в телефонной линии (телефонные компании тоже могут испытывать проблемы).

Наконец, если оканчиваются неудачей попытки разрешения имени сетевого узла, но заранее известен его IP-адрес (который, возможно, был получен раньше) и при использовании пакеты проходят нормально:

```
kalle@tigger:~/projects/rl5> ping 208.201.239.36
PING 208.201.239.36 (208.201.239.36) 56(84) bytes of data.
64 bytes from 208.201.239.36: icmp_seq=1 ttl=46 time=249 ms

--- 208.201.239.36 ping statistics ---
2 packets transmitted, 1 received, 50% packet loss, time 1001ms
rtt min/avg/max/mdev = 249.698/249.698/249.698/0.000 ms
```

то проблемы связаны с разрешением имен в DNS, и дальнейший поиск проблемы следует вести в этом направлении.

traceroute

Команда *traceroute* позволяет продвинуться еще на один шаг вперед по сравнению с утилитой *ping*. Она не только подтвердит возможность связаться с компьютером в Интернете (или в вашей собственной сети), но и покажет, по какому маршруту двигались пакеты. Это может оказаться ценной информацией при диагностике проблем, которые находятся вне зоны вашей ответственности, как, например, проблемы с маршрутизаторами в Интернете – не то чтобы это дало вам возможность решить проблему, но по крайней мере вы будете знать, что с вашей системой все в порядке.

Ниже приводится пример сеанса работы с командой *traceroute*. Обратите внимание: в данном примере мы указали полный путь к команде. Сделано это было потому, что обычно путь к каталогу */usr/sbin* включается в переменную окружения `PATH` только для пользователя *root*, хотя сама команда *traceroute* прекрасно работает, даже если ее запустит обычный пользователь.

```
kalle@officespace:~> /usr/sbin/traceroute www.oreilly.com
traceroute to www.oreilly.com (208.201.239.36), 30 hops max, 40 byte packets
 1 81.169.166.1  0.204 ms  0.174 ms  0.174 ms
 2 81.169.160.157  0.247 ms  0.196 ms  0.195 ms
 3 81.169.160.37  0.351 ms  0.263 ms  0.320 ms
 4 PC1.bln2-g.mcbone.net (194.97.172.145)  0.256 ms  0.273 ms  0.217 ms
```

[1] В лексиконе современных пользователей компьютеров существительное *ping* (свисток, звонок) превратилось в глагол свистеть, звонить.

```
 5 L0.bln2-g2.mcbone.net (62.104.191.140)  0.417 ms  0.315 ms  0.272 ms
 6 lo0-0.hnv2-j2.mcbone.net (62.104.191.206)  4.092 ms  4.109 ms  4.048 ms
 7 lo0-0.hnv2-j.mcbone.net (62.104.191.205)  4.145 ms  4.184 ms  4.266 ms
 8 L0.dus1-g.mcbone.net (62.104.191.141)  8.206 ms  8.044 ms  8.015 ms
 9 c00.ny2.g6-0.wvfiber.net (198.32.160.137)  92.477 ms  92.522 ms  92.488 ms
10 b0-00.nyc.pos1-35-1.wvfiber.net (63.223.28.9) 166.932 ms 167.323 ms 166.356
   ms
11 b00.chi.pos1-6-1.wvfiber.net (63.223.0.214) 167.921 ms 166.610 ms 166.735
   ms
12 63.223.20.53 166.543 ms 166.773 ms 166.429 ms
13 unknown63223030025.wvfiber.net (63.223.30.25) 166.182 ms 165.941 ms 166.042
   ms
14 unknown63223030022.wvfiber.net (63.223.30.22) 165.873 ms 165.918 ms 165.919
   ms
15 unknown63223030134.wvfiber.net (63.223.30.134) 165.909 ms 165.919 ms
   165.832 ms
16 ge7-br02-200p-sfo.unitedlayer.com (209.237.224.17) 165.987 ms 165.881 ms
   166.022 ms
17 pos-demarc.sf.sonic.net (209.237.229.26) 168.849 ms 168.753 ms 168.986 ms
18 0.at-0-0-0.gw4.200p-sf.sonic.net (64.142.0.182) 169.628 ms 0.at-1-0-0.
   gw4.200p-sf.sonic.net (64.142.0.186) 169.632 ms 169.605 ms
19 0.ge-0-1-0.gw.sr.sonic.net (64.142.0.197) 173.582 ms 173.877 ms 174.144 ms
20 gig49.dist1-1.sr.sonic.net (209.204.191.30) 176.822 ms 177.680 ms 178.777
   ms
21 www.oreillynet.com (208.201.239.36) 173.932 ms 174.439 ms 173.326 ms
```

В этом примере попытка трассировки увенчалась успехом, и из полученных результатов видно, сколько времени затрачивалось пакетами на преодоление каждого перехода (путь между соседними маршрутизаторами). При наличии некоторых познаний в области географии и доли воображения можно представить себе маршрут, по которому двигались пакеты. Например, компьютер, на котором была запущена эта команда, географически был расположен в Берлине, Германия[1], таким образом, логично было бы предположить, что имя `bln2` в строках 4 и 5 соответствует некоторому хосту в Берлине, принадлежащему какому-нибудь поставщику интернет-услуг. Посмотрев на карту Германии, далее можно было бы предположить, что переходы с номерами 6 и 7 прошли через Ганновер, а переход с номером 8 – через Дюссельдорф. Вполне очевидно, что дальнейший путь пакетов пролегал по дну Атлантического океана, потому что переходы 9 и 10 с большой долей вероятности пролегали через Нью-Йорк. Очень похоже, что переход с номером 11 прошел через Чикаго, а с 16 по 18 – через Сан-Франциско. Все эти выводы вполне правдоподобны, особенно если учесть, что штаб-квартира O'Reilly (а следовательно, и наиболее вероятное географическое местоположение сервера *www.oreilly.com*/*www.oreillynet.com*) находится в Калифорнии.

dig

dig – это последняя диагностическая утилита, о которой мы поговорим в этом разделе. Утилита *dig* посылает запросы серверам DNS и возвращает информацию о заданном домене. Сразу же начнем с примера:

[1] В то время как сам автор находился в Швеции и благодаря Интернету удаленно зарегистрировался в системе с помощью `ssh`.

```
kalle@tigger:~> dig oreilly.com

; <<>> DiG 9.3.1 <<>> oreilly.com
;; global options: printcmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 52820
;; flags: qr rd ra; QUERY: 1, ANSWER: 2, AUTHORITY: 3, ADDITIONAL: 0

;; QUESTION SECTION:
;oreilly.com.                   IN      A

;; ANSWER SECTION:
oreilly.com.            21600   IN      A       208.201.239.36
oreilly.com.            21600   IN      A       208.201.239.37

;; AUTHORITY SECTION:
oreilly.com.            21600   IN      NS      ns2.sonic.net.
oreilly.com.            21600   IN      NS      ns.oreilly.com.
oreilly.com.            21600   IN      NS      ns1.sonic.net.

;; Query time: 252 msec
;; SERVER: 195.67.199.9#53(195.67.199.9)
;; WHEN: Wed Jul 6 13:31:02 2005
;; MSG SIZE rcvd: 123
```

Какие выводы можно сделать из всего этого? Взгляните на раздел `ANSWER SECTION`. Здесь указаны IP-адреса, используемые серверами DNS, обслуживающими домен *oreilly.com*. Если у вас появятся проблемы с разрешением имен в этом домене, можно попробовать послать ICMP-пакеты по этим адресам, чтобы разобраться, действительно ли проблема обусловлена серверами DNS издательства O'Reilly, или это ваша локальная проблема. В разделе `AUTHORITY SECTION` содержится информация о так называемых авторитетных серверах имен для этого домена, то есть о таких серверах, которые всегда должны давать правильный ответ. Это хорошая практика – иметь по крайней мере два, а еще лучше три сервера DNS в различных сетях, чтобы обеспечить бесперебойную работу службы DNS, даже когда один из серверов останавливается.

Третья строка от конца содержит информацию о сервере имен, который использовался для выполнения запроса к службе DNS. Адрес был взят из параметров настройки локальной системы. Эта информация может использоваться для проверки правильности настроек DNS в вашей системе.

Команда *dig* позволяет указать, какой именно сервер имен должен использоваться для выполнения запроса. Например, если бы у нас появилось желание использовать сервер с адресом 195.67.199.10, тогда команда могла бы выглядеть следующим образом:

```
dig @195.67.199.10 oreilly.com
```

Обычно в таких ситуациях вы получите практически идентичный результат, но если в одном случае вы получаете положительный результат, а в другом нет, тогда, скорее всего, тот сервер, что не ответил, просто был остановлен или недоступен при ваших параметрах настройки сети.

14

Система печати

Эпоха безбумажного общества еще не наступила, хотя ни у кого не вызывает сомнений, что движение в направлении от обычной бумаги к электронным документам будет продолжаться. А пока это время еще не пришло, вам предстоит общаться со многими людьми, перенося электронные документы на отбеленное полотно, полученное из мертвых деревьев.

Поскольку в большинстве дистрибутивов настройка системы печати производится без вашего участия, мы начнем эту главу с обсуждения утилиты командной строки, которая может использоваться для управления принтером и печатью. Затем мы рассмотрим вопросы настройки принтеров, как локальных, так и удаленных, акцентируя основное внимание на простой и мощной универсальной системе печати UNIX (CUPS – Common Printing System UNIX).

Печать

Операционная система Linux предоставляет возможность печатать документы самыми разными способами. Традиционно печать документов выполняется с помощью таких инструментов, как команда `lpr`. Понимание принципов использования этой и других команд формирования и печати документов позволит печатать документы быстро и эффективно. Краткое описание утилит *enscript* и *nenscript* поможет вам создавать красивые распечатки даже из обычных текстовых документов. Программы с графическим интерфейсом предоставляют свои собственные интерфейсы, позволяющие управлять параметрами печати из этих программ. Наконец, мы опишем часть механизмов, лежащих в основе системы печати. Поняв, как все это работает, вы сможете управлять сеансами печати и более эффективно использовать систему печати.

Основные команды печати в Linux

Документ печатается в Linux командой *lpr*. Не обязательно всегда вызывать эту команду непосредственно; можно нажимать кнопку Print (Печать) в каком-нибудь глянцевом графическом интерфейсе, поддерживающем перетаскивание (drag-and-drop). Но в конечном итоге печать будет обрабатываться `lpr` и другими утилитами управления печатью, о которых мы сейчас расскажем.

Если вы хотите распечатать листинг программы, можно ввести команду:

```
$ lpr myprogram.c
```

Другие команды также могут отправлять данные на `lpr` по конвейеру. Программа *lpr* начинает процесс печати, временно записывая данные в каталог, называемый *буфером печати*. Другие части системы управления печатью, о настройке которых мы расскажем в разделе «Управление службами печати» далее в этой главе, изымают файлы из очереди печати в нужном порядке, готовят файлы для печати и управляют потоком данных, отправляемых на принтер.

Для каждого принтера в системе существует по крайней мере один буфер печати, но в любой системе имеется одна очередь печати, которая считается очередью печати по умолчанию. (В старых системах печати LPD очередь по умолчанию традиционно именовалась как `lp`, но это соглашение обычно не действует в системах, где работает система печати CUPS.) Если необходимо указать очередь с другим именем, достаточно воспользоваться параметром `-P` примерно следующим образом: *lpr -Pepson myprogram.c*. Если вы забудете название очереди, посмотрите его в файле */etc/printcap*, воспользуйтесь веб-интерфейсом инструмента настройки CUPS (как описано в разделе «Управление службами печати») или введите команду `lpstat -a`, которая выводит информацию о состоянии всех очередей.

Принтер, предназначенный для работы в различных режимах, например для печати факсов или писем, может иметь отдельные буферы печати для разных целей. Существует возможность создавать дополнительные очереди для одного и того же принтера, чтобы печатать с различным разрешением, на бумаге разного размера и т. п.

Пользователь не видит, подключен ли его принтер непосредственно к компьютеру или находится где-то в сети: все, что вам можно и нужно знать, – это имя очереди печати. Если очередь печати связана с принтером на другой машине, то печатаемый файл сначала помещается в очередь на вашей машине, затем пересылается в соответствующую очередь на машине, к которой подключен принтер, и, наконец, выводится на печать. Подробнее об организации очередей печати рассказывается в разделе «Управление службами печати».

Некоторые программы используют переменную окружения `PRINTER`, с помощью которой определяют, какая очередь печати может использоваться. Поэтому, если вы захотите использовать для большинства задач некоторый конкретный принтер, можно указать его в этой переменной окружения. Например, если в качестве командной оболочки использовать *bash*, можно сделать `epson_360` личным принтером по умолчанию, вставив в файл *.bashrc* команду:

```
$ export PRINTER=epson_360
```

Такой подход срабатывает не для всех программ, поскольку многие из них игнорируют переменную окружения `PRINTER`. Некоторые программы позволяют определять очередь по умолчанию иным способом, например в диалоговом окне. За дополнительной информацией по этой теме следует обращаться к документации, поставляемой вместе с программой. В любом случае, если для печати так или иначе используется очередь `lpr`, название которой может измениться, вы можете использовать параметр *-P*, чтобы указать требуемую очередь. Этот параметр имеет преобладающее значение перед переменной окружения `PRINTER`.

Теперь вы знаете, как напечатать файл, но можете столкнуться с проблемой, когда файл вопреки вашим ожиданиям не печатается сразу. Можно выяснить статус файлов в очереди печати с помощью команды *lpq*. Чтобы выяснить статус файлов, отправленных на принтер, установленный по умолчанию, введите команду:

```
$ lpq
epson_360 is ready and printing
Rank    Owner    Job   File(s)     Total Size
1st     rodsmit  440   (stdin)     2242560 bytes
2nd     rodsmit  441   (stdin)     5199872 bytes
3rd     lark     442   (stdin)     1226752 bytes
```

Вы видите, что принтер работает, но перед вами (если предположить, что вы являетесь пользователем *lark*) в очереди находятся большие задания. Если вам некогда ждать, можно удалить задание из очереди печати. Для удаления задания используйте его номер, сообщенный программой `lpq`:

```
$ lprm 442
```

В результате из буфера печати выбрасывается файл с номером задания 442. Можно сузить отчет *lpq*, запросив сведения только по конкретному номеру задачи (что редко используется), принтеру или идентификатору пользователя. Например, чтобы получить отчет по файлам спулинга, посланным на принтер `hp4500`, следует ввести:

```
$ lpq hp4500
ada is ready and printing
Rank    Owner     Job     File(s)     Total Size
active  lovelac   788     (stdin)     16713 bytes
1st     lark      796     (stdin)     70750 bytes
```

Если вы являетесь пользователем *root*, то можете убить все задания в очереди командой

```
# lprm -
```

Если вы не являетесь пользователем *root*, эта команда убьет только ваши собственные задания. Это ограничение действует и в том случае, если вы укажете принтер:

```
# lprm ada
```

Если вы являетесь пользователем *root*, очередь принтера будет очищена. Если вы являетесь обычным пользователем, из указанного буфера печати будут удалены только те файлы, владельцем которых вы являетесь. Утилита *lprm* сообщает о задачах, которые она убила.

Пользователь *root* может убить все задания печати, созданные каким-либо пользователем, введя команду:

```
# lprm имя_пользователя
```

Команда *lprm* без аргументов удаляет активные задания печати, владельцем которых вы являетесь. Она эквивалентна команде:

```
# lprm ваше_имя_пользователя
```

Если вы хотите узнать, работает ли принтер, выполните команду *lpc*:

```
$ /usr/sbin/lpc status epson_360
```

Подробности вы найдете в разделе «Управление службами печати». Утилита *lpc* обычно устанавливается в каталог */sbin* или */usr/sbin*. Аналогичные задачи в системе печати CUPS призвана решать команда `lpstat`.

Некоторые примеры распространенных задач печати

Иногда возникает необходимость сделать нечто большее, чем просто отправить файл на печать, который уже был подготовлен к выводу на принтер. Например, при желании распечатать страницу справочного руководства или какой-нибудь другой документ, который еще не был подготовлен к печати, необходимо воспользоваться определенными утилитами Linux, зачастую объединенными в конвейер. Например, чтобы быстро получить твердую печатную копию страницы справочного руководства команды `cupsd`, можно дать команду:

```
$ man cupsd | col -b | lpr
```

Команда `man` находит, форматирует и выводит страницу руководства по *cupsd* в формате ASCII с усилением, выражающемся в двойной печати с подчеркиванием символов (вместо курсива) с целью выделения. Выходные данные передаются команде *col* – текстовому фильтру UNIX. Параметр *-b* задает необходимость удаления команд backspace, встроенных в страницу справочного руководства, в результате чего получаются простые текстовые строки с сохранением расположения форматированной страницы руководства. Вывод команды *col* передается *lpr*, которая отправляет текст в каталог буфера печати.

Предположим, вы хотите напечатать страницу руководства с усилением, включающим выделение и прочее. Тогда можно использовать такую команду:

```
$ gunzip -c /usr/share/man/man8/cupsd.8.gz | groff -man -Tps | lpr
```

Команда *gunzip -c* декомпрессирует сжатую страницу справочного руководства и отправит результат на устройство стандартного вывода (точнее, передаст по конвейеру следующей команде). Команда *groff* применяет макрос `man` к указанному файлу, создавая вывод в формате PostScript (что задается параметром *-Tps*). Вывод передается *lpr*, которая отправляет его в буфер печати, а CUPS применяет команды обработки печати, установленные по умолчанию для очереди печати по умолчанию.

Другой полезной утилитой для печати чисто текстовых файлов является команда *pr*, которая форматирует их несколькими способами. Например, с ее помощью можно создать многоколоночный отчет, вывести на печать документ с колонтитулами, пронумеровать строки и многое другое. За дополнительной информацией обращайтесь к страницам справочного руководства команды *pr*.

В большинстве современных дистрибутивов Linux по умолчанию используется система печати CUPS, но в более старых дистрибутивах используется система печати LPRng или еще более старая система BSD LPD. (Некоторые дистрибутивы включают в себя две или все три эти системы печати, но по умолчанию обычно используется CUPS.) Системы печати BSD LPD и LPRng используют команды, аналогичные тем, что описываются здесь. В некоторых системах печати UNIX, таких как SysV, используется другой набор команд, например, для выво-

да файла на печать применяется команда *lp*. Если в вашем дистрибутиве устанавливается такая необычная (для Linux) система печати, вам может потребоваться изучить документацию к ней, чтобы разобраться с тем, как она работает.

Система печати CUPS проектировалась как замена для систем LPD, таких как BSD LPD и LPRng, а также для SysV-подобных систем печати. Как следствие, система CUPS реализует самые основные команды печати, например `lpr` и `lp`, используемые в обеих вышеупомянутых системах.

nenscript и enscript

Утилита *nenscript*, которую теперь часто называют *enscript*, является гибким фильтром, обеспечивающим хорошо форматированный вывод для принтеров PostScript даже из текстовых файлов ASCII. Она не является базовой утилитой Linux, но входит в состав ряда дистрибутивов Linux и может быть загружена с обычных FTP-сайтов Linux.

В принципе, утилиты `nenscript` и `enscript` могут использоваться для вывода содержимого файлов в формате PostScript, но они попрежнему могут использоваться для печати документов на принтерах, не поддерживающих формат PostScript. Как будет описано далее в этой главе, правильно настроенная очередь печати в Linux автоматически выполняет преобразование формата PostScript для вывода на принтеры, не поддерживающие этот формат.

Предположим, что при распечатке программы на языке C вы хотите печатать номера строк, а также выводить текст на бумагу с зелеными полосками (не того формата, который вы используете при печати графики, загружаемой из Интернета, на вашем модном принтере PostScript). Вам нужно обработать текст программы и вставить номера в начале строк. Это можно сделать, обработав файл таким фильтром, как утилита *enscript*. Произведя свою обработку, `enscript` передаст файл `lpr` для помещения в буфер печати и вывода на ваш надежный принтер с непрерывной подачей бумаги (здесь он назван `dino`):

```
$ enscript -C -B -L54 -Pdino -M Letter myprogram.c
```

При наличии параметра *-C* (ранние версии использовали *-N* в этом качестве) фильтр *enscript* выполнит нумерацию всех строк проходящего через него файла. Параметр *-B* подавляет печать обычной заголовочной информации на каждой странице, а параметр *-L54* задает вывод 54 строк на странице. Параметр *-Pdino* просто передается фильтром `enscript` утилите `lpr`, которая интерпретирует его и направляет вывод в каталог буфера печати `dino` для вывода на принтер. Параметр *-M Letter* указывает, что печать будет производиться на листах бумаги формата letter (8,5×11 дюймов). (В зависимости от параметров, заданных при компиляции, утилита `enscript` может по умолчанию использовать формат A4.)

При вызове из командной строки *enscript* автоматически передаст вывод *lpr*, если не задать стандартное устройство вывода с помощью параметра *-p*. Нет необходимости явно переназначать или пересылать вывод по конвейеру на *lpr*.

Допустим, вы собрались сегодня распечатать много программных листингов. Для удобства можно установить переменную окружения для *enscript*, чтобы каждый раз особым образом обрабатывать и печатать ваши листинги:

```
$ export ENSCRIPT=" -C -B -L54 -Pdino -M Letter"
```

Теперь для печати в нужном формате требуется лишь ввести:

```
$ nenscript myprogram.c
```

enscript может посылать вывод в файл, что часто полезно для создания файлов PostScript на системах Linux, фактически не подключенных к принтеру PostScript. Например, чтобы преобразовать текстовый файл в формат PostScript для печати в две колонки на листах формата A4 шрифтом Courier и кеглем 6 пунктов, нужно ввести команду:

```
$ enscript -2 -f Courier6 -M A4 -p document.ps document.txt
```

Параметр `-2` переопределяет печать по умолчанию в одну колонку, а параметр `-f Courier6` переопределяет Courier в 7 пунктов, установленный по умолчанию для вывода в две колонки. (Для печати в одну колонку по умолчанию установлен Courier 10; `enscript` всегда использует шрифт Courier при преобразовании простого текста в PostScript, если не указать обратное с помощью параметра *-f*.) Аргумент `-M A4` задает формат листов бумаги A4. (Чтобы получить полный перечень допустимых параметров, введите команду *enscript --list-media*.) Параметр `-p` указывает, что выходные данные должны быть сохранены в файле *document.ps*, а имя файла без параметра указывает *enscript* на входной файл. Если входной файл не указан, `enscript` использует в качестве входного файла стандартное устройство ввода.

Другой пример. Для печати страницы справочного руководства по *enscript* как базового текста на принтере PostScript введите:

```
$ man nenscript | col -b | enscript
```

Команда `man` извлекает страницу руководства и форматирует ее для вывода текста. Команда *col -b* удаляет инструкции backspace, используемые для выделения и подчеркивания, оставляя простой текст, передаваемый по каналу на фильтр `enscript`. Фильтр преобразует обычный текст в простой формат PostScript с «художественной печатью», который включает верхние и нижние колонтитулы, нумерацию страниц и т. п. Наконец, файл передается *lpr*, которая посылает его в буфер печати. После этого файл обрабатывается системой печати CUPS, как и любой другой файл, то есть файл может быть сразу же отправлен на принтер PostScript, передан Ghostscript или над ним могут быть выполнены какие-либо другие действия.

Если задать *enscript* параметр `-Z`, то он пытается определить, не имеет ли проходящий через него файл формат PostScript, и в таком случае пересылает его дальше без изменений.

Если PostScript-файл передается фильтру `enscript` и принимается им за текстовый файл (возможно, из-за того, что не был задан параметр `-Z`), `enscript` инкапсулирует его и передает на печать. Это может привести к тому, что будет распечатан код PostScript. В этом случае даже маленький файл формата PostScript израсходует уйму бумаги.

Обратите внимание, что принтер, используемый по умолчанию, можно задать в переменной PRINTER или как аргумент *-P*, хранимый в переменной окружения ENSCRIPT. Если указать используемый принтер в переменной ENSCRIPT, то он будет использоваться всякий раз, когда enscript фильтрует один из ваших файлов. Мы рекомендуем устанавливать принтер в переменной PRINTER, а не как аргумент *-P* в переменной ENSCRIPT, благодаря чему можно менять принтер и получать правильную фильтрацию.

Печать с использованием инструментов с графическим интерфейсом

Большинство программ с графическим интерфейсом используют стандартные утилиты печати, такие как *lpr*. В таких программах используются дружественные диалоги печати, как, например, тот, что изображен на рис. 14.1 (этот диалог взят из OpenOffice). Обычно в таких диалогах можно выбрать принтер из раскрывающегося списка (рядом с меткой Name (Имя) на рис. 14.1). Кроме того, вы можете определить другие параметры печати, как, например, номера печатаемых страниц и количество копий. После этого можно щелкнуть на кнопке, запускающей процесс печати, как, например, кнопка OK на рис. 14.1.

Обычно программы с графическим интерфейсом предоставляют не так много возможностей, чтобы откорректировать способ, которым выполняется печать; нет никаких дополнительных параметров, с помощью которых можно было бы использовать иную команду печати, таким образом, вам остается довольствоваться тем, что позволяет сделать программа. Некоторые программы, однако, действительно предоставляют возможность определения дополнительных параметров.

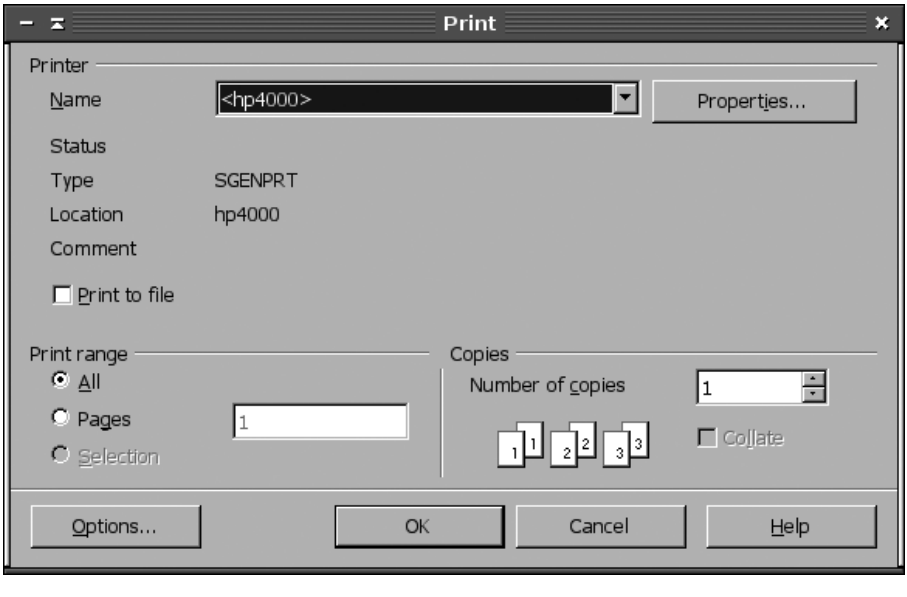


Рис. 14.1. *Типичный диалог печати программ с графическим интерфейсом, использующих традиционные инструменты печати*

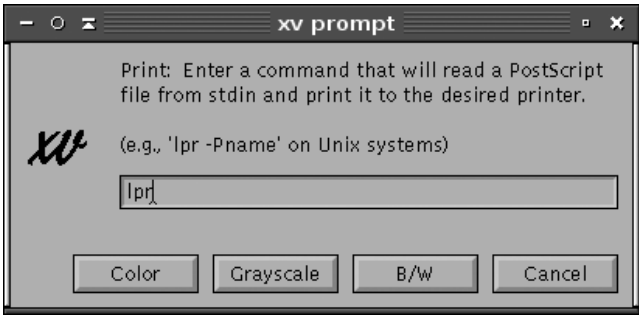


Рис. 14.2. *Некоторые программы дают возможность задать команду и дополнительные параметры печати*

Один из возможных и достаточно очевидных вариантов – это представить диалог печати, где можно ввести команду. Например, на рис. 14.2 изображен диалог печати в программе *xv*. Вместо того чтобы выбирать принтер из раскрывающегося списка, вам нужно ввести команду, включая все необходимые параметры, как если бы команда вводилась из командной строки.

Некоторые программы позволяют выбирать между этими двумя подходами, предоставляя возможность выбирать принтер из раскрывающегося списка или вводить команду печати. Иногда программы могут давать возможность определять команду печати, которую вы хотите использовать, в параметрах настройки программы. Эти параметры, если они имеются, обычно доступны в диалоге настройки личных предпочтений или настройки параметров. Если вы хотите изменить способ, которым выполняется печать в программе, обращайтесь к документации, которая поставляется вместе с программой.

Как система печати обрабатывает файлы

Вообще, после того, как документ или файл будет вручен системе печати, о нем можно забыть до того момента, пока отпечатанные листы не выйдут из принтера. Но когда что-то идет не так, как надо, и отпечатанные листы так и не появляются, или если вы просто любопытны (как и мы!), появляется желание узнать, что происходит на пути между командой `lpr` и выходным лотком принтера. При желании вы можете просто пропустить этот раздел и вернуться сюда позднее.

Только пользователь *root* обладает возможностью прямого доступа к принтеру без использования системы печати. (Впрочем, поступать так будет не очень благоразумно.) Операционная система Linux не дает возможность пользователям обращаться к различным физическим устройствам напрямую, чтобы избежать появления конфликтных ситуаций и нанесения вреда, а также просто потому, что это влечет за собой слишком большой объем ручной работы. Вместо этого утилиты обращаются к фоновым системным процессам, которые ставят задание на печать в очередь. Кроме всего прочего эти процессы выполняют преобразование содержимого исходных файлов, подготавливая его к выводу на определенный принтер, для чего используются язык принтера и протоколы обмена, устанавливается разрешающая способность печати, форматируются страницы, добавляются (или, наоборот, убираются) верхний и нижний колонтитулы и нумеруются страницы.

Во время загрузки Linux сама настраивает себя для работы с физическим устройством, включая настройку портов и протоколов, обслуживающих печать.

Система управления печатью находится под контролем демона *cupsd*, который обладает всеми необходимыми привилегиями, чтобы иметь возможность обращаться к принтерам от имени пользователя. (Устаревшие системы Linux использовали системы печати BSD LPD или LPRng. Демоны этих систем назывались *lpd*, а не *cupsd*, но они решали те же самые задачи.) После установки системы управления печатью демон `cupsd` будет автоматически загружаться при запуске системы. Вся необходимая информация, которая может потребоваться `cupsd` для управления файлами, отправляемыми пользователями на печать, находится в каталоге */etc/cups*.

В системе Linux, где установлена и работает система управления печатью CUPS, имеется два способа печати документов. Первый способ заключается в использовании команд *lpr* и *lp*. Эти программы получили свои имена в честь утилит систем печати BSD (BSD LPD и LPRng) и SysV, соответственно. В качестве аргумента они принимают имя файла, который должен быть напечатан, и ведут себя, во многом напоминая одноименные команды-предшественницы, существовавшие в более ранних системах печати, по крайней мере, для приложений и пользователей, которые к ним обращаются. В действительности же эти программы служат лишь мостиком ко второму способу печати.

В основе второго способа лежат обращения к системным вызовам CUPS. Приложения, использующие этот способ, могут не только посылать, но и получать информацию от CUPS. Кроме передачи демону печати простых текстовых файлов, файлов в формате PostScript или других файлов (как в системах печати старого стиля), приложения могут запросить CUPS предоставить информацию о возможностях принтера. В ответ на это CUPS может вернуть приложению файл в формате PPD (PostScript Printer Definition – описание принтера PostScript). Эти файлы описывают возможности, которыми обладают принтеры PostScript: размеры страницы, размеры области для печати, возможность печати в цвете и т. д. Как будет описано в разделе «Управление службами печати», файлы PPD являются ключевыми компонентами пакетов с драйверами принтеров в Linux даже для принтеров, не поддерживающих формат PostScript. (Для таких принтеров в состав пакета с драйверами включаются файлы PPD, которые описывают их возможности для управления через Ghostscript.) Благодаря возможности двустороннего обмена, программы с поддержкой CUPS могут позволить устанавливать большее число параметров настройки принтера, как, например, разрешающая способность принтера, чего не могут более ранние программы, не имеющие поддержки системы печати CUPS.

Некоторые ранние приложения, не имеющие поддержки CUPS, можно превратить в приложения, обладающие такой поддержкой, изменяя способ печати, который они используют. Так, например, если вместо вызова команды *lpr* вставить вызов команды *kprinter*, то задания по выводу на печать будут обрабатываться системой печати KDE, которая обладает поддержкой CUPS. После этого появится возможность устанавливать дополнительные параметры печати, доступные только в CUPS, например изменять разрешающую способность принтера. Такой способ лучше всего подходит для про-

> грамм с графическим интерфейсом, потому что *kprinter* – это приложение, основанное на возможностях X Window System. Впрочем, этот подход вполне можно использовать для отправки документов на печать из командной строки, например из окна терминала `xterm`.

Совершенно не важно, обладает программа поддержкой CUPS или нет, но после того, как демон *cupsd* примет задание печати, он сохраняет его в каталоге буфера печати (обычно */var/spool/cups*) вместе с файлом, который описывается этим заданием. Затем CUPS упорядочивает список заданий и отправляет их принтеру одно за другим, предотвращая возможность появления конфликтов доступа к принтеру. Каждая очередь печати ассоциируется со своим набором фильтров, которые представляют собой программы, занимающиеся обработкой файлов определенных типов. Параметры настройки фильтров печати глубоко спрятаны в недрах системы от постороннего взгляда и даже от системных администраторов, если, конечно, не слишком углубляться в изучение внутренних механизмов CUPS. Как правило, приложения передают системе печати CUPS простые текстовые документы или документы в формате PostScript. Простой текст может быть передан на принтер без дополнительной обработки, но документы формата PostScript обычно обрабатываются с помощью Ghostscript с целью преобразовать PostScript в формат, понятный принтеру.

Управление службами печати

Система печати в Linux довольно сложна по сравнению со службами печати, имеющимися в большинстве персональных компьютеров. Настройка системы печати обычно выполняется или очень просто (потому что программы настройки автоматически правильно распознают принтер), или очень сложно (когда те же самые программы настройки терпят неудачу просто потому, что принтер плохо поддерживается Linux). На следующих нескольких страницах описывается система печати в Linux. Вначале будет дан краткий обзор программного обеспечения, после чего мы перейдем к настройке и проверке аппаратных устройств, затем настроим универсальную систему печати UNIX (Common UNIX Printing System – CUPS) с помощью инструментальных средств настройки через веб-интерфейс, определим принтеры в CUPS, подстроим некоторые параметры принтера, покажем, как управлять очередями печати, как обеспечить совместимость с более старой системой печати LPD, а также рассмотрим вопросы поиска и устранения неисправностей.

Программное обеспечение для печати в Linux

Печать в Linux основана на взаимодействии нескольких различных пакетов программного обеспечения. Наиболее важный из них – это демон печати, который принимает задания печати, сохраняет их в одной или нескольких очередях и затем отправляет задания принтеру в порядке очередности. Дополнительное программное обеспечение включает в себя: пакет Ghostscript, который выполняет преобразование документов формата PostScript в формат, понятный принтеру; описания принтеров для Ghostscript; а также набор разнообразных инструментов, способных оказать помощь в форматировании документов. Все эти компоненты должны быть установлены, прежде чем можно будет приступать к настройке системы печати.

Демоны печати в Linux

Задача демона печати состоит в том, чтобы, работая в фоновом режиме, принимать от приложений задания печати, сохранять их во временном буфере и передавать их на соответствующие принтеры, избегая при этом появления конфликтных ситуаций. Все основные дистрибутивы Linux включают в себя по крайней мере один демон печати, и в большинстве из них настройки демона (хотя бы минимальные) выполняются в процессе установки операционной системы. Но даже при этом все равно может потребоваться выполнить настройку демона на взаимодействие с конкретным принтером. Решение этой задачи описывается в разделе «Описание принтеров в CUPS».

Традиционно в Linux используется демон Berkeley Standard Distribution Line Printer (BSD LPD) или обновленный пакет программного обеспечения для печати – LPRng. (Для простоты изложения обе системы будут далее упоминаться как системы LPD.) Основу системы LPD составляют довольно простые инструменты; они принимают задания печати, сохраняют их в очередях и затем отправляют их непосредственно на принтер. Эти системы могут быть использованы для передачи заданий печати через конвейер другим программам дополнительной обработки, если в этом есть необходимость. В отличие от систем печати для Windows, Mac OS и других операционных систем, система печати LPD не обеспечивает возможности двустороннего обмена информацией с приложением. Например, приложение не может запросить у системы LPD сведения о ширине страницы или возможности печати цветных изображений. Таким образом, необходимо настраивать каждое приложение, чтобы сообщить им об особенностях принтера. Тем не менее система LPD обладает поддержкой сетевых возможностей, что позволяет нескольким компьютерам совместно использовать один принтер.

В 1999 году появилась новая экспериментальная система печати – CUPS. В основу этого пакета был заложен новый сетевой протокол печати и предусмотрена возможность для приложений запрашивать и получать информацию о возможностях принтера, а также определять дополнительные параметры печати, что было невозможно с более ранними системами LPD. К 2004 году все основные дистрибутивы Linux либо полностью переключились на использование CUPS, либо предоставили возможность выбора системы печати наряду с BSD LPD или LPRng. По этой причине в данной главе описывается именно система CUPS. Хотя некоторые принципы и программная поддержка CUPS в равной степени относятся к BSD LPD и LPRng, но глубинные механизмы этих систем совершенно различны. Если вы до сих пор пользуетесь одной из ранних систем печати, попробуйте обдумать вопрос перехода на использование CUPS.

В большинстве случаев установить CUPS (если она еще не установлена) можно с помощью инструментальных средств управления пакетами, входящими в состав дистрибутива. Для этого нужно отыскать пакет с названием `cups` и установить его. Если система уже настроена на работу с BSD LPD или LPRng, тогда сначала необходимо удалить этот пакет. При желании можно загрузить оригинальные исходные тексты CUPS с веб-страницы проекта *http://www.cups.org*.

Ghostscript

Традиционная система LPD отправляет файл, полученный от приложения, на принтер. При такой организации приложение должно само создавать файл в том

формате, который будет понятен принтеру. Это существенно отличает систему LPD от систем печати в других операционных системах, таких как Windows, где приложение может обратиться к операционной системе за помощью в подготовке документа к печати. Как показывает практика, приложения UNIX и Linux в большинстве случаев генерируют на выходе файлы двух форматов:

Простой текст

Программы могут отправлять на принтер файлы в простом текстовом формате исходя из предположения, что в качестве принтера используется типичное устройство построчной печати (то есть быстрый принтер с некоторыми возможностями печати форматированного текста) или принтер по крайней мере способен воспринимать простой текст.

PostScript

Язык Adobe PostScript – это один из многих языков принтеров. Он обрел популярность в 1980-х годах с появлением лазерных принтеров. Большинство приложений Linux, которые позволяют печатать документы, содержащие сложное форматирование текста, различные шрифты и графические изображения, практически всегда на выходе создают файл формата PostScript.

К сожалению, большинство принтеров, особенно недорогие модели, которые приобретаются для использования дома или в малом бизнесе, не понимают PostScript. В связи с этим файл, полученный от приложения, должен пройти обработку в Ghostscript (*http://www.cs.wisc.edu/~ghost/*). Ghostscript – это интерпретатор PostScript, который может размещаться в компьютере, а не в принтере. Ghostscript преобразует PostScript в форматы, понятные большинству принтеров. Фактически комбинацию PostScript и Ghostscript в Linux можно рассматривать как эквивалент драйвера системы печати в Windows.

Несмотря на то, что CUPS в корне изменила представление о системе печати, она до сих пор во многом основана на применении Ghostscript для преобразования файлов формата PostScript на языки, понятные принтерам. По этой причине, если предполагается использовать принтер, который не понимает формат PostScript, необходимо устанавливать Ghostscript. К счастью, все основные дистрибутивы включают в себя пакет Ghostscript. Чтобы проверить, был ли установлен этот пакет в вашей системе, поищите пакет с именем `ghostscript`.

Пакет Ghostscript распространяется фактически в виде двух версий. Самая свежая версия пакета Ghostscript – это пакет AFPL Ghostscript, распространяемый на основе лицензии, которая допускает бесплатное использование в разных целях, но запрещает бесплатное распространение. Спустя несколько месяцев после выхода версии AFPL Ghostscript она выходит уже под лицензией GPL как GNU Ghostscript, и именно эта версия поставляется в составе большинства дистрибутивов Linux. В большинстве случаев несколько месяцев отставания от самых передовых нововведений в Ghostscript – это не такой большой срок. Если по каким-то причинам вам совершенно необходимо иметь самую последнюю версию, посетите домашнюю страницу проекта Ghostscript.

Пакет Ghostscript поставляется вместе с драйверами большинства распространенных принтеров и способен генерировать на выходе множество графических форматов файлов. С помощью Ghostscript можно даже создавать документы в формате PDF (Adobe Portable Document Format – переносимый формат документов).

(Этот процесс может быть автоматизирован с помощью сценария на языке командной оболочки `ps2pdf`.) Для достижения большей гибкости можно добавить в систему драйверы Ghostscript различных принтеров.

Так как Ghostscript представляет для нас определенный интерес, следует заметить, что данный пакет рассматривает все принтеры как графические устройства. Таким образом, при выводе на печать обычного текстового документа Ghostscript преобразует текст в растровый рисунок и отправляет этот рисунок принтеру. Это означает, что Ghostscript не может использовать шрифты, встроенные в принтер. Это также означает, что печать документов через Ghostscript иногда выполняется более медленно, чем печать тех же документов на том же принтере через другое программное обеспечение, такое как драйверы печати в Windows. (Обычно этот эффект незначителен, но в некоторых случаях замедление бывает просто ужасающим.) На некоторых очень старых лазерных принтерах проявляется другой негативный эффект – Ghostscript (а следовательно, и Linux) требует, чтобы в принтере имелась дополнительная память при использовании полной разрешающей способности принтера. Впрочем, Windows также воспринимает многие принтеры исключительно как графические устройства, поэтому различия в подходах к печати можно считать несущественными.

Описания принтеров

Стандартная установка CUPS поддерживает довольно узкий спектр принтеров, обычно это модели с поддержкой PostScript и некоторые принтеры компаний Hewlett-Packard и Epson. Чтобы получить поддержку других принтеров, необходимо установить пакет драйверов принтеров. (По правде говоря, многие из этих «драйверов» на самом деле представляют собой просто описания принтера, которые воспринимаются стандартными драйверами Ghostscript; тем не менее они необходимы для обеспечения возможности работы с принтером.) Существуют несколько таких пакетов драйверов:

Foomatic

В этот пакет включены драйверы, которые можно найти на сайте *http://www.linuxprinting.org/foomatic.html* для большинства популярных принтеров. При поисках подходящего драйвера для своего принтера сюда следует обращаться в первую очередь.

GIMP Print

Программа обработки изображений GNU (GNU Image Manipulation Program, GIMP) – это пакет для работы с графикой, получивший широкую известность, который включает в себя собственный набор драйверов принтеров. Они оформлены в виде отдельного пакета драйверов принтеров, который может использоваться совместно с CUPS (а также с системами печати BSD). Драйверы пакета GIMP Print представляют собой отличный выбор в случае необходимости выводить на печать графические изображения. Дополнительную информацию о пакете можно найти на сайте *http://gimp-print.sourceforge.net*.

ESP Print Pro

Разработчиками CUPS был создан ряд описаний принтеров, доступных на коммерческой основе. Подробнее узнать об этом можно на сайте *http://www.easysw.com/printpro*.

В состав большинства дистрибутивов входят пакеты Foomatic и Gimp Print, поэтому можно попробовать поискать эти пакеты в своем дистрибутиве. Иногда этим пакетам дают разные имена и нередко включают в них дополнительные наборы описаний принтеров. Если попытка установки каких-либо описаний принтеров потерпит неудачу, то в процессе настройки системы печати будет доступен лишь ограниченный набор принтеров.

Иногда для одного и того же принтера имеется несколько описаний, например, разные описания одного и того же принтера могут поставляться в составе пакетов Foomatic и GIMP Print или даже упаковываться в один и тот же пакет. В таких случаях можно опробовать все доступные имеющиеся описания, чтобы выяснить, какое из них лучше всего работает с вашим принтером и с типами документов, которые обычно приходится печатать. Иногда разработчики делают подсказки, выделяя предпочтительное описание в списке.

Дополнительное программное обеспечение печати

Система печати CUPS (или LPD) совершенно необходима для обеспечения возможности печати в Linux. Для большинства принтеров не менее необходимы пакет Ghostscript и описания драйвера Ghostscript. В некоторых случаях хотя и не обязательно, но желательно иметь следующее дополнительное программное обеспечение:

`enscript` ***или*** `nenscript`

Эти команды были описаны ранее в этой главе. Они помогают преобразовывать простой текст в PostScript и другие форматы, что может оказаться очень удобным.

`groff`

Эта программа является свободно распространяемой реализацией системы набора roff, которая позволяет создавать красиво отформатированные документы из текстовых файлов с использованием языка разметки `troff/nroff`. Многие современные пользователи предпочитают использовать для этих целей текстовые процессоры с графическим интерфейсом, тем не менее `groff` по-прежнему может использоваться различными инструментальными средствами и находить себе применение как самостоятельная утилита.

TEX и LATEX

Пакет TEX представляет собой систему создания документов высокого качества, а пакет LATEX является его расширением. Этот набор инструментальных средств выбирается многими математиками, учеными, программистами и инженерами для создания своих документов. Часть документации к Linux поставляется в формате TEX или LATEX, хотя эти же документы обычно доступны и в других форматах.

Lout

Пакет Lout также достоин внимания как эффективное и компактное средство форматирования документов для вывода в PostScript. Он поддерживает Level 2 PostScript и соглашения Adobe по структуризации (Adobe Structuring Conventions), требует относительно немного памяти и поставляется с хорошей документацией, позволяющей быстро его освоить. Lout не создает промежу-

точного выходного формата; он преобразует входную разметку непосредственно в выходной PostScript.

`netpbm` *и* `pbmplus`

Эти программы поддерживают преобразования большого количества различных графических форматов файлов. (Такие форматы требуется преобразовывать в PostScript, прежде чем выводить на печать.)

Ghostview

Этот пакет содержит инструменты, позволяющие просматривать файлы формата PostScript в среде X Window System, а также предоставляет поддержку форматов PostScript и PDF для других пакетов, например веб-броузера.

ImageMagick

Данная программа позволяет выводить в окне X большое количество графических форматов и осуществлять их взаимное преобразование. (При необходимости вывести изображение в формате PostScript он использует Ghostview и Ghostscript.) Большинство графических файлов, которые можно напечатать, можно также вывести на экран с помощью ImageMagick.

Пакеты поддержки факсимильных устройств

Если нужна поддержка факсимильных устройств, то утилиту *tiffg3* можно использовать с Ghostscript для выдачи факсимильных файлов в формате Group III. Для управления под Linux факс-модемом Class 1 или Class 2 можно воспользоваться пакетом `efax`, который поставляется в составе многих дистрибутивов, либо можно установить и настроить более мощные, но и более сложные пакеты FlexFax или HylaFax.

Некоторые из этих инструментов описываются в данной книге. Многие из них требуют незначительной настройки, и все они являются инструментальными средствами пользовательского уровня. Этот список далеко не окончательный, для Linux существует огромное число программного обеспечения печати, начиная от инструментов форматирования простого текста и заканчивая сложными текстовыми процессорами и прикладными программами для работы с графикой.

Настройка устройств печати

Прежде всего, суть настройки принтера заключается в настройке аппаратных средств. Для начала необходимо убедиться, что принтер совместим с Linux, проверить физическое соединение с компьютером и убедиться в его исправности. Неисправность соединительного кабеля и портов ввода-вывода может привести к появлению проблем при настройке принтера в Linux, превратив ваши попытки в погоню за несбыточным, потому что невозможно решать проблемы при неисправном оборудовании.

Проверка совместимости принтера

Основная проблема, связанная с совместимостью принтера, – это язык, используемый принтером. Ряд языков получили наиболее широкое распространение, но во многих принтерах используются свои собственные уникальные языки. Из числа наиболее распространенных языков можно назвать PostScript, язык управления принтерами Hewlett-Packard (Printer Control Language, PCL) и язык Epson

ESC/P2. (ESC/P2 – это наиболее распространенный язык, использующийся в матричных принтерах.) Многие изготовители производят принтеры, которые используют все перечисленные языки, но иногда они забывают указать это в своих описаниях или упоминают эти языки совсем в другом контексте. Типичный пример – лазерный принтер, который использует PCL; производитель может позиционировать свой принтер как HP-совместимый (обычно со ссылкой на определенную модель принтера Hewlett-Packard).

Лучший признак совместимости с Linux – это наличие поддержки формата PostScript в принтере. Обычно к таким принтерам относятся лазерные принтеры среднего и высшего класса. Хотя существуют недорогие струйные модели, обладающие поддержкой PostScript. При наличии поддержки PostScript нет нужды беспокоиться о существовании драйвера Ghostscript для принтера; достаточно будет просто настроить CUPS на передачу «сырого» формата PostScript прямо на принтер.

Некоторые принтеры рекламируются как поддерживающие PostScript, хотя фактически такая поддержка в них отсутствует. Изготовители идут на такой шаг, когда вместе с принтером поставляют свое программное обеспечение, напоминающее Ghostscript, но, как правило, это программное обеспечение предназначено для работы в операционной системе Windows. Таким образом, если в наличии имеется принтер с объявленной поддержкой PostScript, прежде всего, необходимо убедиться, что эта поддержка реализована в самом принтере, а не в пакете с драйверами для Windows.

Если принтер не обладает поддержкой PostScript, тогда, чтобы убедиться в совместимости с Linux, лучше всего посетить сайт Linux Printing, точнее, его базу данных принтеров, на странице *http://www.linuxprinting.org/printer_list.cgi*. Отыщите свою модель принтера в раскрывающихся списках на этой странице и щелкните на кнопке Show. После этого в окне броузера появится описание принтера и замечания о его совместимости с Linux. В описании может также указываться, где найти драйверы или описания принтера, используемые системой CUPS, благодаря чему можно убедиться в наличии соответствующей программной поддержки.

Если на сайте Linux Printing принтер помечен как «paperweight» («пресс-папье», то есть данная модель вообще не работает под Linux), можно попробовать обратиться к поисковым системам и поискать в Интернете упоминания названия принтера вместе со словом «Linux». Если очень повезет, можно таким образом наткнуться на недавно появившийся или экспериментальный драйвер и попробовать его использовать. В противном случае, вероятно, придется поискать другой принтер, более дружественный по отношению к Linux.

Если рассматривается возможность приобретения многофункционального устройства (например, устройства, выполняющего функции сканера и принтера), необходимо убедиться в наличии поддержки Linux для каждой из функций.

Иногда случается так, что принтер в многофункциональном устройстве прекрасно работает, но сканер «оживить» так и не удается. Как правило, поддержка отдельных функций таких устройств обеспечивается отдельными проектами, такими как Ghostscript для принтеров и Scanner Access Now Easy (SANE) для сканеров. Иногда проект создается специально для того, чтобы сосредоточить все не-

обходимые драйверы в одном месте, как, например, проект HP Office Jet (*http://hpoj.sourceforge.net*) или проект, где находятся драйверы для продукции компании Epson (*http://www.avasys.jp/english/linux_e*). Эти проекты могут быть как независимыми, так и получать поддержку изготовителей устройств.

Интерфейсы принтеров

Принтеры могут подключаться к компьютеру разными способами. Наибольшее распространение получили четыре следующих способа подключения:

Параллельный порт

Этот способ подключения используется очень широко. В большинстве компьютеров на базе процессора x86 имеется единственный параллельный порт, специально предназначенный для подключения принтера (хотя существуют и другие устройства, разрабатывавшиеся также для подключения к этому порту). Параллельный порт обладает неоспоримым преимуществом в скорости перед последовательным портом RS-232. В Linux параллельному порту обычно соответствует устройство */dev/lp0*, хотя ему может быть назначено и устройство */dev/lp1*, и даже устройство с еще большим порядковым номером, особенно если в компьютер были добавлены дополнительные платы адаптеров параллельного порта для подключения нескольких принтеров.

Последовательный порт RS-232

Некоторые устаревшие модели принтеров подключаются к последовательному порту RS-232 компьютера. Скорость обмена информацией по последовательному порту существенно ниже по сравнению с параллельным, именно по этой причине последовательный порт постепенно стал выходить из употребления в последние десятилетия. Если принтер подключается к последовательному порту, скорее всего, доступ к нему можно будет организовать через устройство */dev/ttyS0* или */dev/ttyS1*, хотя возможны устройства с большими порядковыми номерами. К последовательному порту часто подключаются и другие устройства, такие как мышь или модем, поэтому, возможно, придется внимательно изучить, к какому разъему какой кабель подключается, чтобы определить, какое из устройств соответствует принтеру. Возможно также, придется воспользоваться программой `setserial` для настройки последовательного порта RS-232.

Порт USB

В последние годы порт универсальной последовательной шины (Universal Serial Bus, USB) вытеснил параллельный порт как более предпочтительный для подключения принтера. Стандарт USB 1.x практически сравнялся по скорости обмена с параллельным портом, а USB 2.0 превзошел его в этом, таким образом, интерфейс USB обеспечивает более высокую скорость печати. В последних версиях ядра доступ к принтерам, подключенным к порту USB, выполняется с помощью дерева каталогов */proc/bus/usb*, а система CUPS должна автоматически обнаруживать принтеры USB при условии, что к моменту запуска демона CUPS принтер соединен с компьютером и включен. Так как CUPS обычно автоматически запускается при загрузке системы, это означает, что принтер желательно подсоединять к компьютеру и включать перед загрузкой системы.

Ethernet

Некоторые принтеры поддерживают возможность подключения к Ethernet прямо или косвенно. Принтеры для рабочих групп часто имеют порты Ethernet, и их можно обнаружить в сети, как если бы они были обычными компьютерами. Кроме того, существуют специализированные серверы печати. Эти устройства позволяют подключать принтеры через параллельный порт или порт USB, превращая их в сетевые принтеры, подключенные к Ethernet. Принтеры этого класса поддерживают один или более сетевых протоколов печати. Наилучшим вариантом, с точки зрения Linux и CUPS, считается, если принтер поддерживает протокол печати для Интернета (Internet Printing Protocol, IPP), который используется системой печати CUPS. Менее предпочтительными являются протоколы LPD и Server Message Block/Common Internet File System (SMB/CIFS – блок серверных сообщений/общая файловая система Интернета).

Перед подключением принтера необходимо определить, к какому типу интерфейса он подключается. Некоторые типы проверок и параметры настройки не будут работать с некоторыми интерфейсами, например, будет не так просто в обход системы печати послать текстовый файл непосредственно на принтер Ethernet или USB. Кроме того, чтобы подключить принтер, необходимо иметь соответствующие порты на компьютере и соединительные кабели. Все, чего не хватает, придется приобрести. Например, можно приобрести платы расширения с любым типом порта. Если на компьютере присутствуют только разъемы USB, можно приобрести адаптер, с помощью которого подключить принтер с параллельным интерфейсом, с интерфейсом RS-232 или даже Ethernet к порту USB. Правда, при этом следует проверить наличие драйвера адаптера для Linux!

Если принтер поддерживает различные способы подключения, наилучшим выбором будет интерфейс USB, затем в порядке убывания предпочтений следуют параллельный порт и за ним последовательный. Интерфейс USB обеспечивает возможность получения более подробной информации о принтере, использует более тонкий соединительный кабель и (в версии 2.0) дает более высокую скорость обмена, чем параллельный порт. Наличие возможности подключения принтера к Ethernet может стать большим плюсом, если предполагается организовать доступ к принтеру с нескольких компьютеров, но если принтер использует необычный протокол, это может принести гораздо больше неприятностей, чем выгоды, особенно если принтер будет использоваться только в одной системе. Кроме того, обязательно следует учесть наличие и доступность других интерфейсов и соединительных кабелей, например, можно столкнуться с серьезными проблемами при попытке установить карту USB в старую систему, которая используется в качестве сервера печати.

Проверка подключения принтера

Прежде чем начать настройку служб печати, необходимо убедиться в наличии соединения с устройством печати. Если у вас на машине установлена еще какая-нибудь операционная система, например Microsoft Windows, проверьте с ее помощью правильность подключения и работы печатающих устройств, прежде чем загружать Linux. Если вам удалось напечатать документ из другой операционной системы, то тем самым сразу устраняется один из главных источников пере-

живаний и головной боли. Аналогичным образом, если вы собираетесь использовать сетевые службы печати, то предварительно нужно проверить подключение к сети и функционирование всех сетевых протоколов.

Если принтер подключен к параллельному или последовательному порту, можно попробовать отправить какой-нибудь документ прямо в файл устройства, соответствующий принтеру. Например, проверить подключение принтера к параллельному порту можно следующей командой:

```
# cat /etc/fstab > | /dev/lp0
```

Эта команда скопирует содержимое файла */etc/fstab* в устройство */dev/lp0* – обычно это устройство соответствует параллельному порту. Если принтер может печатать текстовые файлы, результатом этой команды должна быть распечатка вашего файла */etc/fstab*. Возможно, что на принтере мигнут индикаторы состояния, но больше ничего не произойдет. Иногда нажатие кнопки (с надписью Form Feed, Continue или чем-то подобным) на корпусе принтера может помочь распечатать файл. Такое может произойти, потому что принтер получил неполную страницу данных и ждал появления дополнительных данных. При работе в обычном режиме такого не происходит, но при выполнении проверки принтера вполне возможно. Еще одна проблема – печать «лесенкой» – возникает, когда при печати не вставляется символ возврата каретки в конце каждой строки, как в следующем примере:

```
/dev/hda1 / ext3 defaults 1 1
                              /dev/hda5 /home ext3 defaults 1 2
```

Можно выяснить, почему это происходит. В текстовых файлах UNIX каждая строка заканчивается символом новой строки (или перевода строки, код ASCII равен 10). В MS-DOS символы новой строки и возврата каретки используются в паре. Следовательно, ваш принтер настроен на конец строки в стиле MS-DOS с обоими символами в конце каждой строки. На практике система CUPS обычно фильтрует символы перевода строки, чтобы принтер, настроенный под формат MS-DOS, мог правильно интерпретировать конец строки. Однако, если проблема остается даже после окончательной настройки системы печати, можно попробовать перенастроить принтер таким образом, чтобы он возвращал каретку при получении символа перевода строки. Обычно это делается установкой микропереключателей в соответствующее положение. Посмотрите инструкцию к вашему принтеру. (Будьте осторожны при изменении характеристик принтера, если вы используете несколько операционных систем.)

Если используется принтер, подключенный к порту USB, найти его можно командой `lsusb`:

```
$ lsusb
Bus 005 Device 004: ID 04b8:0807 Seiko Epson Corp.
```

На практике команда `lsusb` дает немного больший объем информации, здесь же демонстрируется лишь та часть результатов, которая относится к принтеру USB – Epson RX500. В данном случае принтер подключен к шине USB с номером 5, устройство 4. Получить более подробную информацию можно при просмотре содержимого файла */proc/bus/usb/devices*, однако для непосвященных эти сведения будут малопонятны. Один из параметров, который следует отыскать в данном

файле, — это `Driver=usblp`. Если такая строка в файле присутствует, значит, операционная система Linux распознала принтер, а отсюда следует, что CUPS должна обеспечить возможность взаимодействия с принтером. (Впрочем, способность выводить документы на принтер во многом зависит от наличия и состояния драйверов Ghostscript.)

Наконец, принтеры Ethernet могут быть проверены точно так же, как любые другие сетевые устройства. Вне всяких сомнений, можно воспользоваться утилитой `ping`, чтобы проверить наличие соединения принтера с сетью. Более серьезные проверки таких принтеров зависят от сетевых протоколов, которые понимает принтер. Хотя существует возможность низкоуровневой диагностики этих протоколов, самый простой способ состоит в том, чтобы попробовать настроить принтер, как будет описано далее в разделе «Описание принтеров в CUPS».

Настройка системы безопасности CUPS

Перед продолжением обсуждения темы настройки CUPS необходимо рассмотреть некоторые вопросы обеспечения безопасности CUPS и параметры настройки инструментальных средств, доступных через Веб. В отличие от систем LPD, системой печати CUPS лучше всего управлять с помощью инструментов, доступных через Web. Однако иногда доступ к этим средствам заблокирован или настроен не совсем правильно. (Некоторые дистрибутивы предлагают свои собственные инструментальные средства с графическим интерфейсом для настройки CUPS. Если установлен такой дистрибутив, можно использовать имеющиеся инструментальные средства или выполнять настройки с помощью стандартного набора инструментальных средств CUPS.) В зависимости от параметров сетевого окружения может возникнуть желание включить или отключить возможность *просмотра*, которая позволяет серверам CUPS обмениваться списками доступных принтеров друг с другом. Возможность просмотра существенно упрощает настройку и обслуживание сетевых принтеров, но в некоторых дистрибутивах она выключена по умолчанию. С другой стороны, при необходимости строгого соблюдения режима безопасности или при непосредственном подключении системы к Интернету было бы желательно отключить возможность просмотра, но в некоторых дистрибутивах она, наоборот, включена по умолчанию. В любом случае желательно просмотреть значения этих параметров настройки и убедиться, что они установлены в соответствии с текущими потребностями.

Включение возможности настройки через Веб

Все основные настройки CUPS хранятся в файле */etc/cups/cupsd.conf*. Этот файл смоделирован по образу и подобию файла с настройками веб-сервера Apache, таким образом, если вы знакомы с Apache, в этом файле вы должны чувствовать себя как дома. Для тех, кто незнаком с файлом настроек Apache, мы поясним: файл начинается с ряда глобальных директив, которые оформлены в виде пар имя–значение. Например, чтобы изменить имя сервера по умолчанию (отправляемое другим системам), можно ввести директиву:

```
ServerName gutenberg.example.com
```

Данная строка определяет имя сервера как *gutenberg.example.com*. Система печати CUPS обладает большим количеством возможных директив, полный список которых можно получить на страницах справочного руководства. Отдельные

группы директив могут объединяться в разделы, ограниченные строками, заключенными в угловые скобки:

```
<Location /admin>
AuthType Basic
AuthClass System
Order Deny,Allow
Deny From All
Allow From 127.0.0.1
Allow From 192.168.1.0/24
</Location>
```

Такой набор директив применяется только к определенным функциям сервера, в данном случае директивами описывается, кому разрешен доступ к подсистеме `/admin` (администрирования). Эту особенность можно использовать для выборочной настройки системы безопасности (или других функций) CUPS. Чтобы иметь возможность администрирования CUPS через сеть, очень важно организовать безопасный доступ к подсистеме `/admin`. Директивам, которые были продемонстрированы в предыдущем примере, отвечают параметры, которые следует настроить в первую очередь:

`AuthType`

Директива `AuthType` определяет тип аутентификации, который должен использоваться для организации доступа к заданной функции. В случае с подсистемой `/admin` обычно используется тип `Basic`, при котором пароли передаются по сети в открытом виде. Если возникнет необходимость удаленного администрирования, значение этой директивы желательно изменить на `Digest`, в этом случае пароли будут передаваться в зашифрованном виде. (При этом необходимо будет составить список паролей с помощью программы *lppasswd*.) Для доступа к некоторым подсистемам аутентификация обычно не используется, для таких подсистем директива `AuthType` должна иметь значение `None`. Данные настройки обычно устанавливаются по умолчанию и позволяют пользователям обращаться к принтерам без необходимости ввода паролей.

`AuthClass`

Данная директива определяет, какие группы пользователей могут иметь доступ к подсистеме. Возможные значения: `Anonymous`, `User`, `System` и `Group`. Параметр `Anonymous` указывает, что аутентификация пользователей не должна производиться, `User` — что любой пользователь при вводе корректного имени и пароля должен получить доступ к подсистеме, `System` — что доступ к подсистеме получают только те пользователи, которые входят в состав системной группы CUPS. (В разных дистрибутивах это могут быть разные группы, но в большинстве случаев это группы *sys*, *system* или *root*.) Значение `Group` дает возможность определить имя группы пользователей в директиве `AuthGroupName`, которая должна стоять в следующей строке.

`Order`

Данная директива определяет порядок предоставления доступа к CUPS по умолчанию. Когда она имеет значение `Deny,Allow`, все попытки будут отвергаться, если право на доступ не будет оговорено явно. В случае значения `Allow,Deny` доступ будет предоставляться, если иное не будет оговорено явно.

`Deny` *и* `Allow`

Данные директивы описывают машины или сети, откуда доступ должен быть запрещен или разрешен, соответственно. Здесь можно использовать IP-адреса хостов и адреса сетей, имена хостов, имена доменов (начинающиеся с символа точки), ключевые слова `All` или `None`, а также переменные `@IF` (за которыми должно следовать имя сетевого интерфейса в круглых скобках) и `@LOCAL` (для обозначения всех локальных сетей).

Чтобы иметь возможность пользоваться веб-инструментами настройки CUPS, необходимо определить в файле настроек раздел `/admin` и разрешить доступ к подсистеме с адреса 127.0.0.1, как показано в предыдущем примере. В этом примере административный доступ предоставляется также хостам из сети с адресом 192.168.1.0/24. Вообще говоря, подобную возможность следует активировать, только если система выступает в роли сервера печати, который предполагается администрировать удаленно, поскольку она увеличивает риск вторжения через возможные бреши в системе безопасности CUPS.

Если доступ к веб-инструментам настройки CUPS должен быть закрыт (например, потому, что в системе имеются дополнительные инструменты настройки), следует удалить все директивы `Allow` и добавить директиву `Deny From All`.

Включение и выключение возможности просмотра

В процессе изучения содержимого файла */etc/cups/cupsd.conf* можно проверить функцию просмотра списка принтеров. В контексте CUPS термин «просмотр» означает автоматический поиск принтера в сети. Данная возможность поддерживается на уровне протокола IPP и позволяет серверам, работающим по этому протоколу, обмениваться списками принтеров друг с другом. Каждый сервер периодически отправляет широковещательные посылки, на которые откликаются другие серверы IPP. Благодаря этому настройка принтера должна производиться всего один раз, на сервере, к которому этот принтер подключен. После этого сервер отсылает информацию о параметрах своего принтера другим серверам CUPS/IPP. Таким образом, новый принтер станет доступен приложениям на удаленных системах, после того как эти приложения будут перезапущены. Данная возможность дает существенную экономию времени на настройке принтера, особенно в сети, где принтеры часто то исчезают, то появляются.

По поводу возможности обмена списками принтеров хотелось бы сделать два замечания. Первое: как и любая другая сетевая служба, функция просмотра несет определенное бремя безопасности. Системы, настроенные на автоматический поиск принтеров, легко обмануть, предоставив им поддельную информацию о принтерах, а при наличии возможных недочетов в программном коде CUPS это может привести к достаточно серьезным негативным последствиям. По этой причине в некоторых дистрибутивах данная возможность запрещена по умолчанию. Этот факт влечет за собой второе наше замечание: если функция просмотра действительно необходима, ее следует разрешить. В случае, когда функция просмотра отключена, система не сможет обнаруживать принтеры автоматически. Аналогично, если к системе подключены принтеры, которые предполагается предоставить в общее пользование, придется включить определенные параметры настройки. Таким образом, необходимо будет проверить значения некоторых параметров как на стороне клиента, так и на стороне сервера:

`BrowseAllow` *и* `BrowseDeny`

Эти директивы указывают CUPS на стороне клиента адреса, от которых должны приниматься или отвергаться, соответственно, пакеты функции просмотра. Данные директивы имеют тот же формат, что и директивы `Allow` и `Deny`. В локальных сетях установка директивы `BrowseAllow @LOCAL` обычно дает достаточно хорошие результаты, позволяя системе автоматически обнаруживать удаленные принтеры. В таком варианте данный параметр сообщает системе печати, что она может принимать пакеты просмотра, поступающие со всех локальных адресов. В этой директиве можно указывать IP-адреса или имена хостов, чтобы повысить уровень безопасности, или определять адреса целых сетей, ослабляя или изменяя уровень безопасности. Можно также явно исключить отдельные компьютеры или сети с помощью директивы `BrowseDeny`. Раздел может содержать множество директив `BrowseAllow` и `BrowseDeny`.

`Browsing`

Данная директива может иметь значения `On` или `Off`. Если предполагается, что сервер CUPS должен предоставлять свои принтеры в общее пользование, в эту директиву следует записать значение `On`, которое является значением по умолчанию в типовых настройках, поставляемых в составе пакета CUPS. (В некоторых дистрибутивах этот параметр по умолчанию выключен.)

`BrowseAddress`

Данная директива предоставляет возможность указать серверу CUPS, по каким адресам можно рассылать пакеты просмотра. С ее помощью можно определить IP-адреса хостов или сетей и имена хостов в том же виде, что и для директив `Allow` и `Deny`. По умолчанию никакие пакеты не посылаются, таким образом, в случае необходимости этот параметр придется настроить. Для небольших локальных сетей подойдет директива `BrowseAddress @LOCAL`, но, возможно, потребуется добавить в файл с настройками несколько таких директив или настроить систему иным образом, в зависимости от имеющегося сетевого окружения.

Обычно эти директивы указываются в глобальном разделе файла *cupsd.conf*. Однако иногда они могут дополнительно указываться в разделе `/printers`, поэтому при наличии проблем следует проверить и этот раздел.

Фактически, чтобы иметь возможность печатать на принтере, на стороне сервера необходимо организовать возможность доступа к разделу `/printers` независимо от того, разрешена или запрещена функция просмотра. В связи с этим на серверах CUPS обычно имеется одна или более директив `Allow` в разделе `/printers`. Без этих директив (в этом или в глобальном разделе) сервер CUPS будет отклонять поступающие задания печати.

Перезапуск CUPS

После внесения изменений в файл с настройками демон CUPS должен быть перезапущен. В большинстве дистрибутивов это можно сделать с помощью сценария SysV начального запуска:

```
# /etc/init.d/cupsd restart
```

Данная команда (или похожая на нее – путь к сценарию `cupsd` может изменяться в зависимости от дистрибутива) остановит демон системы печати CUPS и затем снова запустит его. В результате в силу вступят все изменения, которые были внесены в файл с настройками.

Описание принтеров в CUPS

Теперь, когда доступ к инструментам администрирования открыт, можно приступить к их использованию. Для этого потребуется обычный веб-броузер. Подойдет любой современный броузер: Mozilla Firefox, Konqueror, Opera и даже веб-броузер текстового режима Lynx. С помощью веб-броузера можно добавлять описания новых принтеров и выполнять проверку настроек.

Здесь описывается порядок администрирования CUPS с помощью встроенного веб-интерфейса, потому что этот подход одинаков для всех дистрибутивов Linux. Во многих дистрибутивах имеются собственные средства администрирования системы печати. Например, в дистрибутивах Fedora и Red Hat имеется отдельный инструмент настройки принтера (`system-config-printer`), в дистрибутиве SUSE для этих целей используются YaST и утилиты YaST2. При желании можно воспользоваться этими инструментальными средствами. Они позволяют выполнять настройки тех же параметров, что и утилита настройки, входящая в состав CUPS, но порядок их использования несколько отличается.

Доступ к инструменту описания принтера

Доступ к средствам настройки CUPS выполняется через порт с номером 631, поэтому, чтобы приступить к настройкам, в адресной строке броузера необходимо ввести адрес *http://localhost:631*. Если предполагается администрировать удаленную систему печати, чтобы обратиться к инструменту настройки этой системы, необходимо в адресной строке указать ее сетевое имя, например *http://gutenberg.example.com:631*.

Если административный доступ к системе печати ограничен адресом 127.0.0.1 (`localhost`), попытки обращения к инструментам настройки по обычному имени хоста будут отвергнуты, даже если они производятся с того же самого компьютера. Если доступ к подсистеме администрирования ограничивается, необходимо в строке броузера указывать имя хоста `localhost` или IP-адрес 127.0.0.1.

Если CUPS была настроена должным образом и запущена, в результате обращения к порту компьютера 631 в окне веб-броузера должна появиться страничка с различными ссылками, такими как: Do Administration Tasks (Выполнение административных задач), Manage Printer Classes (Управление классами принтеров), On-Line Help (Оперативная справка) и т. д. Если броузер сообщает об ошибке, возможно, что-то было настроено неправильно. В этом случае попробуйте еще раз перечитать предыдущий раздел «Настройка системы безопасности CUPS» и проверить, работает ли демон `cupsd`. Если все в порядке и страничка появилась в окне броузера, можно приступить к настройке принтеров.

Создание описания принтера

Чтобы добавить принтер, необходимо щелкнуть на ссылке Manage Printers (Управление принтерами) на главной странице настройки CUPS. В результате должна быть открыта страница, напоминающая рис. 14.3, где изображены некоторые удаленные принтеры, обнаруженные системой CUPS автоматически.

Чтобы создать описание нового принтера, нужно:

1. Убедиться, что принтер подсоединен к компьютеру и включен. Если принтер подсоединяется к порту USB, но еще не был подсоединен или включен, следует перезапустить демон CUPS после подсоединения и включения принтера, как это описывалось выше в разделе «Перезапуск CUPS».
2. Щелкнуть на ссылке Add Printer (Добавить принтер). Эта ссылка находится внизу страницы и потому не видна на рис. 14.3. Если перед этим процедура аутентификации еще не была пройдена, на экране появится окно, где будет предложено ввести имя пользователя и пароль. Здесь следует ввести имя пользователя *root* и его пароль. Если процедура аутентификации будет пройдена благополучно, в окне броузера появится страница, озаглавленная как Add New Printer (Добавление нового принтера), где следует ввести основные сведения о принтере: название, размещение и краткое описание.
3. Названия даются принтерам для их идентификации в приложениях и обычно выбираются не очень длинные, например `lexmark` или `hp4500`. Лучше использовать названия, состоящие из одного слова (или нескольких слов, разделенных символом подчеркивания). Названия, содержащие символы дефиса, могут приводить к ошибкам в работе CUPS, поэтому таких имен следует избегать.

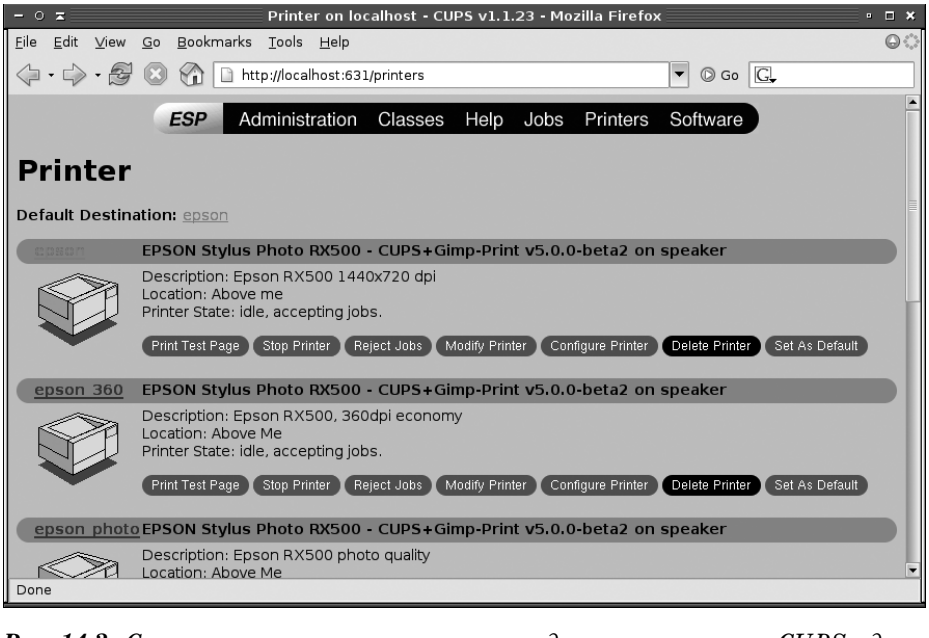


Рис. 14.3. *Страница инструмента сетевого администрирования CUPS, где перечислены все принтеры, включая автоматически обнаруженные на других системах*

Поля, содержащие указание на местоположение и описание, служат исключительно описательным целям, поэтому сюда можно вводить любую информацию, которая поможет отличать данный принтер от других. Когда все поля будут заполнены, следует щелкнуть на ссылке Continue (Продолжить).

4. После этого будет открыта страница Device (устройство), где необходимо определить устройство, используемое для связи с принтером. В раскрывающемся списке будет предложено выбрать одно из следующих устройств:
   a. Параллельный порт, идентифицируемый как Parallel Port #1 (Параллельный порт #1), Parallel Port #2 (Параллельный порт #2) и т. д.
   b. Последовательный порт RS-232, идентифицируемый как Serial Port #1 (Последовательный порт #1), Serial Port #2 (Последовательный порт #2) и т. д.
   c. Принтер USB, идентифицируемый как Printer USB #1 (Принтер USB #1), Printer USB #2 (Принтер USB #2) и т. д. Кроме того, в скобках должны появиться название фирмы-производителя и модель принтера. Если этого не произошло, следует проверить соединение принтера с компьютером и включить питание принтера, после чего перезапустить CUPS и повторить процесс с самого начала.
   d. Сетевые принтеры будут идентифицированы протоколами обмена, такими как LPD/LPR Host or Printer (Компьютер или сетевой принтер, поддерживающий протокол LPD/LPR), Windows Printer via SAMBA (Подключение к принтеру Windows через протокол SAMBA) или Internet Printing Protocol (Протокол печати для Интернета). Тип поддерживаемого протокола должен быть известен заранее, и в зависимости от выбранного протокола на врезке Printing to Network Printer (Печать на сетевой принтер) можно будет определить некоторые дополнительные параметры.
   e. Помимо вышеуказанных вариантов в списке будут присутствовать и другие, используемые достаточно редко.

   Выбрав устройство для связи с принтером, нажмите Continue.

5. После этого должна появиться страница Model/Driver (Модель/драйвер), как показано на рис. 14.4. Здесь нужно выбрать модель принтера из списка и затем щелкнуть на ссылке Continue (Продолжить). Обратите внимание: в зависимости от того, какой пакет драйверов был установлен, в списке один и тот же производитель может встретиться дважды, возможно, с несколько разным написанием, например HP и Hewlett-Packard. В этом случае следует выбрать сначала первый вариант и, если в списке моделей требуемый принтер отсутствует, вернуться назад и попробовать выбрать второй вариант. Если в списке отсутствует некоторый производитель, причем один из известных, таких как Epson, Canon или Lexmark, следует установить пакет с описаниями принтеров, как это было описано выше в разделе «Описания принтеров». Если сложно определить производителя принтера, например, потому, что он мало известен в мире, следует обратиться к руководству по эксплуатации принтера, чтобы выяснить, какие модели эмулирует имеющийся принтер, и выбрать одну из них. Некоторые общераспространенные языки принтеров, включая PostScript, PCL и ESC/P2, присутствуют в пункте Generic (Обычный), таким образом, если сложно определить тип имеющегося принтера, но известно, что он

использует один из общераспространенных языков, можно попробовать выбрать этот пункт.

6. За первой страницей Model/Driver (Модель/драйвер) последует вторая страница с таким же названием, где можно выбрать модель принтера, например Lexmark Optra Color 45 или Lexmark Z51. После того как выбор будет сделан, нужно щелкнуть на ссылке Continue (Продолжить). Если в списке требуемая модель отсутствует, нужно щелкнуть на кнопке Back (Назад) в панели инструментов броузера и попробовать выбрать другое название фирмы-производителя или же выбрать другую, совместимую модель. Если модель принтера так и не удалось отыскать ни в одном из списков, следует установить новые или дополнительные определения принтеров; в этом вам поможет веб-сайт Linux Printing, где можно попытаться отыскать свою модель принтера.
7. После щелчка на ссылке Continue (продолжить) на второй странице Model/Driver (Модель/драйвер) CUPS должна ответить сообщением о том, что принтер был добавлен. Если теперь щелкнуть на ссылке Printers (Принтеры) вверху страницы, снова появится страница со списком принтеров (см. рис. 14.3), но на этот раз в списке должен присутствовать только что добавленный принтер. Теперь можно выйти и выполнить более тонкие настройки принтера.

Если будет необходимо уточнить некоторые параметры настройки принтера, необходимо щелкнуть на ссылке Modify Printer (Изменить принтер) в области с описанием принтера на главной странице со списком принтеров (см. рис. 14.3). После этого процедура настройки будет проходить по тем же страницам, только на этот раз параметры настройки будут иметь значения, установленные ранее. (Вы не сможете изменить имя принтера, но для изменения будут доступны остальные параметры.)

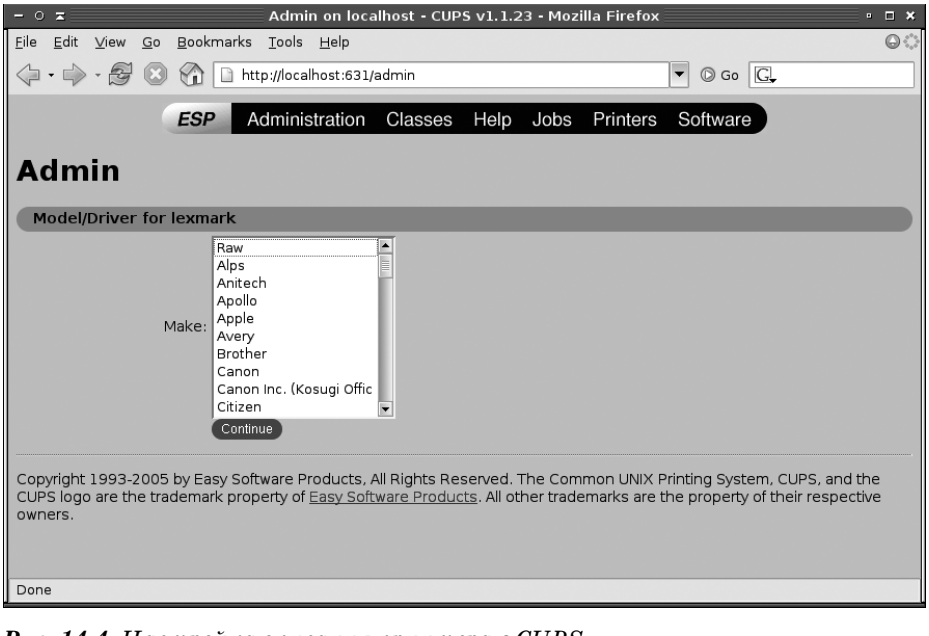


Рис. 14.4. *Настройка описания принтера в CUPS*

Данная процедура (которая, впрочем, может быть выполнена с помощью инструментов с графическим интерфейсом, входящих в состав дистрибутива) представляет собой простейший способ настройки очереди печати в CUPS. Хотя, если все параметры принтера известны заранее, можно те же самые действия выполнить с помощью утилиты *lpadmin*. При определении новой очереди печати используется следующий синтаксис обращения к утилите:

```
lpadmin [-E] [-h сервер] -p принтер параметры...
```

Параметр *-E* активизирует очередь печати (обычно именно это и требуется), параметр *-h* используется для настройки удаленного сервера, а параметр *-p* определяет название очереди.

Заключительная часть команды – самая интересная. Из наиболее часто указываемых параметров используются *-i интерфейс*, который определяет интерфейс устройства, и *-m модель*, который указывает на файл формата PPD (PostScript Printer Definition – описание принтера PostScript), описывающий данную модель принтера и его возможности. Имя файла PPD, которое может быть недостаточно очевидным, если он не был получен, например, от фирмы-производителя, для заданного принтера должно быть известно заранее. Поскольку само название формата PPD подразумевает описание возможностей принтера, поддерживающего PostScript, изготовители принтеров обычно создают подобные файлы только для моделей с поддержкой PostScript. Описания принтеров CUPS основываются на файлах PPD для любых типов принтеров, но фактически большая часть файлов PPD распространяется в составе пакетов Foomatic, GIMP Print и других.

Один из типов очереди печати требует отдельного упоминания – это *«сырая» очередь* (*raw queue*). Для создания очереди этого типа нужно выбрать пункт Raw («Сырой») в списке производителей и Raw Queue («Сырая» очередь) в списке моделей принтеров. В отличие от большинства типов очередей принтеров CUPS, при использовании «сырой» очереди фильтрация системой печати не выполняется, то есть CUPS не будет пытаться определить тип печатаемого файла, не будет передавать его другим программам, таким как Ghostscript, чтобы преобразовать документ в вид, который сможет быть воспринят принтером. В большинстве ситуаций «сырая» очередь не имеет практической ценности, однако в некоторых случаях необходимость в такой очереди вполне может возникнуть. Первый пример: если предполагается использовать Linux в качестве сервера печати для операционных систем, отличных от Linux, таких как Windows. Тогда на Windows-клиентах можно было бы установить соответствующие драйверы и настроить их так, чтобы печать выполнялась в «сырую» очередь на Linux. Использование «сырой» очереди в таком применении дает гарантию, что CUPS не будет изменять задания печати Windows. (Одна из разновидностей такого сценария использования, когда драйверы PostScript устанавливаются на компьютерах с операционной системой Windows. Каждый из этих подходов имеет свои плюсы и минусы.) Второй пример, где «сырая» очередь может найти применение, это случай, когда используется приложение, предоставляющее собственные драйверы принтера. Одно из таких приложений – GIMP, при использовании драйверов принтеров, поставляемых в составе этого приложения, можно добиться лучших результатов, чем при использовании стандартных драйверов Ghostscript.

В случае каких-либо сомнений, вероятно, лучший выход – создать обычную очередь печати, которая будет выполнять предварительное преобразование докумен-

Печать на сетевых принтерах

Процедура описания сетевых принтеров требует указания большего числа параметров, чем для локальных (как было описано на шаге 4 раздела «Создание описания принтера»). В любом случае необходимо будет на странице Device URL (Адрес устройства), где находится единственное поле ввода, ввести строку, идентифицирующую удаленный принтер. Точное представление строки зависит от используемого протокола обмена:

- Для принтеров LPD строка адреса должна начинаться с префикса `lpd://`, за которым должны следовать имя хоста и имя очереди печати. Например, строка `lpd://gutenberg/lexmark` описывает принтер `lexmark`, подключенный к компьютеру `gutenberg`.
- Для принтеров, обслуживаемых операционной системой Windows или Samba, доступ к которым организован по протоколу SMB/CIFS, имя хоста и имя принтера указываются так же, как и в случае с принтерами LPD, но при этом необходимо дополнительно указать имя пользователя и пароль. Например, полная строка адреса принтера в сети:

  ```
  smb://printacct:ppass@GUTENBERG/LEXMARK
  ```

  описывает принтер `LEXMARK`, подключенный к компьютеру `GUTENBERG`, для доступа к которому используются имя пользователя `printacct` и пароль `ppass`. В зависимости от настроек сервера возможно, что имя пользователя и пароль указывать не потребуется.
- Лучший способ настройки принтеров IPP состоит в том, чтобы использовать функцию просмотра CUPS. Если это невозможно, тогда для указания адреса принтера может использоваться префикс `ipp://`, за которым указываются имя хоста и имя принтера точно так же, как и в случае с принтерами LPD. Некоторые сетевые устройства, обладающие поддержкой протокола IPP, требуют указания дополнительных параметров, поэтому перед выполнением настроек обязательно следует ознакомиться с руководством по эксплуатации устройства.

тов (с помощью Ghostscript и других программ-фильтров) в формат, распознаваемый принтером. Если такой подход не даст положительных результатов или заранее известно об этом, можно попробовать использовать «сырую» очередь.

Проверка описания принтера

После того как очередь печати будет создана, ее можно протестировать. Для этого на главной странице описания принтера (см. рис. 14.3) нужно щелкнуть на ссылке Print Test page (Напечатать пробную страницу) в описании требуемой очереди. В ответ на это CUPS должна ответить сообщением, что страница отправлена на печать. После короткой задержки принтер должен отпечатать пробную страницу. (Точная длительность задержки устанавливается экспериментальным путем и зависит от модели принтера, параметров настройки, типа подключения к компьютеру, быстродействия самого компьютера и степени загруженности операционной системы.)

Стандартная пробная страница CUPS включает в себя круговую палитру цветов, круг, состоящий из радиальных линий (для демонстрации разрешающей способности), информацию о размерах страницы, границах, номинальном значении разрешения и об интерпретаторе PostScript. Если степень разрешения (число, выводимое в области Imageable Area, выражается в точках на дюйм, или dpi) не устраивает, прочитайте раздел «Дополнительная настройка описаний принтеров». Точно так же, если цветной принтер выводит изображение в градациях серого цвета, необходимо подкорректировать параметры настройки. (Здесь предполагается, что принтер заведомо исправен, так как иногда бывают ситуации, когда цветные чернила в принтере закончились или забиты дюзы цветной печатающей головки!)

Если принтер вообще ничего не напечатал, прочитайте раздел «Устранение неполадок в системе печати» далее в этой главе.

Дополнительная настройка описаний принтеров

Большинство современных принтеров предоставляют дополнительные параметры настройки, которые влияют на качество печати. В список этих параметров попадают: разрешающая способность, режимы экономии чернил или тонера, режимы печати, оптимизированные для бумаги различного типа и т. п. Система печати CUPS предоставляет возможность изменять значения по умолчанию, принятые для этих параметров в данной очереди. Чтобы выполнить дополнительные настройки, нужно на главной странице веб-интерфейса CUPS (см. рис. 14.3) щелкнуть на ссылке Configure Printer (Настроить принтер), расположенной в области описания требуемого принтера. В результате откроется список параметров, как показано на рис. 14.5, которые можно будет изменить.

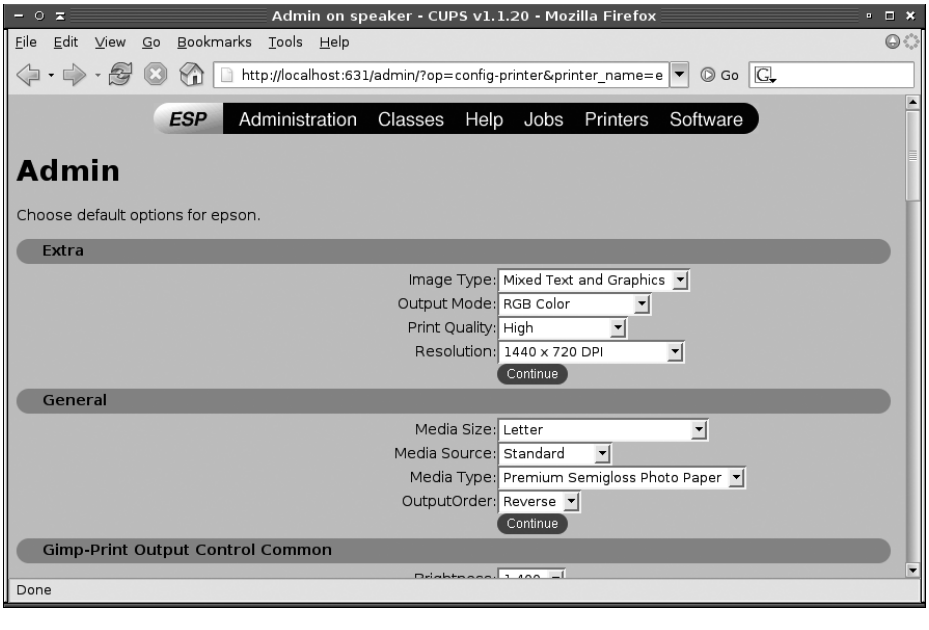


Рис. 14.5. *Система печати CUPS предоставляет возможность дополнительной настройки любой очереди печати из имеющихся*

Список доступных параметров может изменяться в зависимости от типа принтера. Из наиболее общих параметров, которые может потребоваться изменить, в списке можно найти следующие:

Output Mode (Режим печати)

Этот параметр определяет режим печати цветных изображений: должны ли они печататься как цветные изображения или как черно-белые, и если как цветные, то как должна декодироваться информация о цвете. Для большинства цветных принтеров лучшим выбором будет значение RGB Color.

Print Quality (Качество печати)

Этот параметр описывает разрешающую способность печати в условных единицах (Economy (Экономичный режим), Standard (Стандартный режим), High (Высокое качество) и т. д.).

Resolution (Разрешение)

С помощью этого параметра можно установить разрешающую способность более точно, в числовом выражении. Многие принтеры поддерживают возможность устанавливать отдельно вертикальную и горизонтальную разрешающие способности. Главное правило, которое следует помнить при выборе разрешающей способности, – чем выше разрешающая способность, тем ниже скорость печати, особенно это проявляется в струйных принтерах.

Media Size (Размер бумаги)

С помощью этого параметра можно определить размер бумаги. Этот параметр обязательно следует проверить, поскольку иногда, при начальной установке принтера, размер бумаги по умолчанию выбирается неправильно.

Media Source (Источник бумаги)

Многие принтеры поддерживают возможность подачи бумаги из нескольких лотков, и этот параметр позволяет определить, какой из них должен использоваться по умолчанию.

Media Type (Тип бумаги)

Этот параметр позволяет указать тип используемой бумаги. Эта особенно важно для струйных принтеров, потому что различные типы бумаги по-разному впитывают чернила. Когда CUPS заранее знает, какой тип бумаги используется, она может настроить параметры вызова Ghostscript для более качественной печати.

Output Order (Порядок печати)

В зависимости от конструктивных особенностей, принтеры могут печатать многостраничные документы как в прямом, так и в обратном порядке. Этот параметр может использоваться для ликвидации проблемы, когда документ печатается не в том порядке, в каком требуется.

Duplexing and paper handling (Двусторонняя печать)

Некоторые драйверы позволяют включать режим двусторонней печати (когда печать выполняется с двух сторон листа бумаги) и изменять другие дополнительные параметры, имеющие отношение к работе с листами бумаги.

Color and brightness options (Параметры управления цветом и яркостью)

Многие параметры имеют отношение к настройкам яркости, насыщенности и баланса цветов в цветных принтерах. Этот набор может включать в себя самые разные параметры, в зависимости от используемой модели принтера. Возможно, придется поэкспериментировать с этими параметрами, чтобы добиться наилучших результатов.

Dithering options (Параметры цветопередачи)

Программа Ghostscript позволяет использовать различные алгоритмы цветопередачи, то есть, когда программа дает принтеру команду напечатать какой-то определенный цвет, последний, скорее всего, попытается смешать чернила различных цветов. Причина проста: цветные принтеры имеют в своем распоряжении ограниченное число цветов, и, таким образом, чтобы вывести некоторый определенный цвет, будет предпринята попытка смешать два или более цветов. Для одних применений лучше работает один алгоритм смешивания цветов, для других – другой, а кроме того, существенную роль играют личные предпочтения человека. Таким образом, CUPS предоставляет возможность сообщить программе Ghostscript, какой алгоритм цветопередачи следует использовать. То же относится и к параметрам настройки яркости: возможно, придется немного поэкспериментировать, чтобы добиться желаемых результатов.

Specialty print options (Специализированные параметры)

Некоторые принтеры поддерживают возможность печатать без границ (без пустых областей по краям страницы), на дисках CD-R или DVD-R и другие специализированные возможности. Если драйвер Ghostscript поддерживает эти параметры, их можно будет изменить в соответствии со своими потребностями.

Print direction and interleaving options (Направление печати и параметры чередования)

Большинство современных струйных принтеров поддерживают двунаправленную печать, то есть печать одной строки производится в одном направлении, а другой – в другом, что позволяет сократить время печати. Это очень удобно, но иногда качество выравнивания улучшается при печати только в одном направлении. То же касается и параметров чередования, которые управляют порядком использования дюз печатающих головок с целью улучшения или снижения качества печати, что может влиять на скорость печати.

Ink options (Чернила)

Некоторые драйверы позволяют указывать, какие чернильницы следует использовать, или сообщать программе Ghostscript, какие дополнительные наборы чернил были установлены.

Banners (Баннеры)

Существует возможность заставить CUPS печатать страницу с баннером до или после выполнения задания печати. Данная страница может нести идентификационную информацию о задании печати и краткое примечание, как, например, слово `Confidential` (Секретно).

PostScript level filtering (Уровень фильтрации PostScript)

Для некоторых принтеров, обладающих поддержкой PostScript, CUPS может выполнять преобразование уровня PostScript до определенной версии, например, преобразовывать все до версии PostScript Level 1. Это может оказаться необходимым, когда система зависает при использовании более свежих версий PostScript.

Современные принтеры обладают таким количеством параметров настройки, что описать здесь их все не представляется возможным. Возможно, вам придется самостоятельно поэкспериментировать с некоторыми из параметров и на собственном опыте разобраться с тем, какое влияние они оказывают на конечный результат. Некоторые параметры большее влияние оказывают на текст, другие на графику. Разные параметры могут по-разному влиять на разные типы графических изображений, например на цифровые фотографии и диаграммы. Но, как правило, если вы не понимаете, для чего служит тот или иной параметр, то лучше оставить его в покое.

В прошлом Linux не позволяла приложениям изменять параметры настройки принтеров из тех, что были описаны выше, или предоставляла им ограниченные возможности. Фактически приложения, ориентированные на использование старой системы печати LPD, не в состоянии изменять эти параметры непосредственно, они могут лишь использовать принтеры с настройками по умолчанию. Чтобы установить определенные значения параметров настройки для таких приложений, приходится создавать несколько очередей печати для каждого из принтеров. Например, допустим, что необходимо печатать на одном и том же принтере с разрешением 360 точек на дюйм (для быстрой печати) и с разрешением 1440 точек на дюйм для получения высококачественных документов. В этом случае необходимо создать две очереди печати (например, `canon_360` и `canon_1440`). Обе очереди должны подключаться к одному и тому же устройству связи с принтером, но с разными параметрами по умолчанию. После этого, чтобы получить документ с тем или иным качеством печати, его можно отправить в ту или в другую очередь.

Как вариант в графической оболочке X можно воспользоваться программой `kprinter`, если приложение позволяет ввести команду печати. В результате при попытке распечатать документ из приложения будет выводиться стандартный диалог печати KDE. В этом диалоге можно выбрать систему печати CUPS (в списке в нижней части окна диалога) и очередь печати, а затем щелкнуть на кнопке Properties (Свойства) в верхнем правом углу диалогового окна. В результате появится диалог Configuration (Настройка), как показано на рис. 14.6. В этом диалоге имеется несколько вкладок, которые соответствуют различным параметрам настройки CUPS, благодаря чему можно настроить разрешение (рис. 14.6), а также другие параметры печати.

Еще один дополнительный параметр настройки – выбор очереди печати по умолчанию. Теоретически этот параметр можно настроить через веб-интерфейс CUPS, щелкнув на ссылке Set as Default (Установить по умолчанию), но на практике такие попытки часто приводят к появлению сообщений об ошибке выполнения неизвестной операции. Чтобы обойти эту ошибку, можно воспользоваться командой *lpadmin* с параметром *-d*, в котором передается название очереди по умолчанию:

```
# lpadmin -d hp4500
```

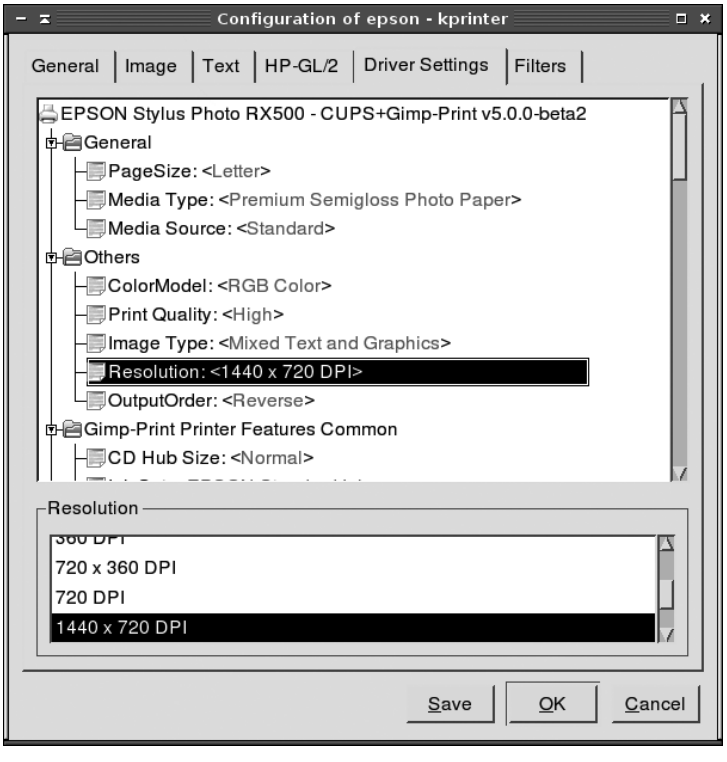


Рис. 14.6. *Программа kprinter позволяет выполнять настройки печати даже для приложений, не поддерживающих CUPS*

В данном примере эта команда определяет очередь с именем `hp4500` как очередь печати по умолчанию. После этого все задания печати, отправляемые командой *lp* или *lpr*, для которых не указана конкретная очередь, будут отправляться на печать в очередь `hp4500`. Эта очередь также будет выбираться по умолчанию при первом запуске диалога печати. Однако сохраняется возможность отправки заданий в очереди печати не по умолчанию как из командной строки, так и из приложений с графическим интерфейсом, для чего надо лишь указать требуемую очередь явно.

Управление очередями печати

Важной частью администрирования принтеров является управление активными очередями печати. Как только в системе появляются очереди печати, пользователи начинают работать с ними, в результате чего иногда могут возникать некоторые проблемы. Иногда будет возникать необходимость удалять слишком большие задания, временно закрывать очередь на время ее перенастройки или как-то иначе управлять очередями. Для решения подобных задач CUPS предоставляет два класса инструментов: инструменты командной строки и инструменты на основе веб-интерфейса.

Инструменты командной строки

Команды CUPS позволяют управлять очередями печати из командной строки, например из окна *xterm* или любого другого инструмента текстового режима. Большинство из этих команд для своей работы требуют наличия привилегий суперпользователя, но некоторые из них могут быть исполнены обычными пользователями:

lpc

Данная команда унаследовала свое имя от менеджера очередей печати в системе LPD, но версия команды для CUPS имеет значительные ограничения. Единственный полезный параметр команды – *status*. В результате исполнения команды *lpc* `status` на экран будут выведены список имеющихся очередей и сведения о состоянии каждой из них (постановка в очередь разрешена, печать разрешена, количество заданий в очереди и доступность очереди для демона CUPS). Чтобы получить сведения о какой-то одной очереди, команде следует передать имя этой очереди.

lpstat

Эта команда выводит информацию о заданиях печати и, так же как команда *lpc*, может вызываться обычными пользователями. Если ввести эту команду без параметров, она выведет список всех заданий печати, ожидающих обработки во всех очередях печати, какие имеются в системе. Команда имеет ряд дополнительных параметров, которые влияют на содержимое выводимой информации. Например, с параметром *-p имя-очереди* команда выведет сведения только для заданной очереди, с параметром *-u пользователи* будут выведены сведения, касающиеся заданий печати, созданных только указанным пользователем или пользователями, а с параметром *-d* будет выведена информация только об очереди, используемой по умолчанию.

lpq

Как и *lpc*, данная команда заимствована из системы печати LPD. Она выводит информацию о состоянии текущей очереди примерно так же, как это делает команда *lpstat*. Она часто вызывается с параметром *-P имя_очереди*, чтобы вывести сведения об очереди, отличной от очереди по умолчанию. (Параметр *-a* может использоваться для вывода информации обо всех очередях.)

lppasswd

Данная программа изменяет файл со списком паролей CUPS, то есть файл, используемый системой печати в случае, когда в файле настройки *cupsd.conf* в параметр `AuthType` записано значение `Digest`.

enable и disable

Эти команды запускают и останавливают очередь, соответственно. Обе команды принимают имя очереди в качестве параметра, а команда *disable* может принимать еще несколько дополнительных параметров, самым важным из которых может считаться *-c*. Данный параметр отменяет обработку всех заданий печати, находящихся в текущий момент времени в очереди. Кроме того, команда *disable* может принимать параметр *-r причина*, где строка *причина* описывает причину остановки очереди.

Если будет получено сообщение об ошибке `not a shell builtin` (не встроенная команда оболочки) при вызове команды *enable*, значит, необходимо указать полный путь к команде, как правило, это будет */usr/bin/enable*.

accept и reject

Эти команды открывают или закрывают прием новых заданий в указанную очередь печати. По своему действию эти команды отличаются от команд *enable* и *disable* тем, что *accept* и *reject* оказывают влияние на прием заданий в очередь, тогда как *enable* и *disable* влияют на возможность печати заданий в очереди. Подобно *disable*, команда *reject* имеет параметр `-r причина`, чтобы регистрировать причину отказа приема новых заданий.

lprm

С помощью этой команды можно удалить задание из очереди. Команда может принимать имя очереди (`-P имя_очереди`) и идентификатор задания (который можно узнать с помощью команды *lpstat*) в виде дополнительных параметров.

lpmove

Данная команда перемещает задание печати из одной очереди в другую. Она принимает два параметра: идентификатор задания и имя очереди, например *lpmove oldqueue-456 newqueue*. В данном случае команда переместит задание *oldqueue-456* в очередь *newqueue*.

С целью демонстрации использования этих команд предположим, что возникла некоторая проблема с очередью `hp4000`, возможно, принтер неисправен и требует технического обслуживания. В связи с этим возникла необходимость закрыть очередь, одни задания, находящиеся в ней, переместить в другую очередь (например, `laserwriter`), а некоторые другие – удалить. Чтобы выполнить все эти действия, можно было бы дать следующие команды:

```
# disable hp4000
# lpc status hp4000
hp4000:
        printer is on device '/dev/null' speed -1
        queuing is enabled
        printing is disabled
        no entries
        daemon present
# lpstat
hp4000-433              sholmes        15360   Sun Apr 10 22:20:44 2005
hp4000-434              moriarty        2977   Sun Apr 10 22:21:32
# lpmove hp4000-433 laserwriter
# lprm hp4000-434
# lpstat
laserwriter-433         sholmes        15360   Sun Apr 10 22:20:44 2005
```

В этом примере сначала была запрещена печать заданий из очереди `hp4000`, и команда *lpc status hp4000* подтвердила это сообщением `printing is disabled`. Следующая команда, *lpstat*, продемонстрировала, что в заблокированной очереди осталось два невыполненных задания. Далее первое задание, принадлежащее пользователю `sholmes`, перемещается в очередь `laserwriter` командой *lpmove*, а задание, принадлежащее пользователю `moriarty`, удаляется командой *lprm*. После-

дующая проверка с помощью команды *lpstat* показывает, что задание действительно было перемещено. (Это задание не удалось бы обнаружить, если бы оно было полностью отправлено на принтер до того, как была введена команда.)

Веб-интерфейс CUPS

Веб-интерфейс CUPS предоставляет альтернативную возможность управления очередями, а возможности этого инструмента ничуть не уступают возможностям утилит командной строки. Находясь в основном списке принтеров (см. рис. 14.3), можно остановить печать заданий в очереди (эквивалентно команде *disable*) или закрыть очередь для приема новых заданий (эквивалентно команде *reject*), щелкнув на ссылке Stop Printer (Остановить принтер) или Reject Jobs (Закрыть прием заданий), соответственно. После щелчка текст ссылок меняется на Start Printer (Запустить принтер) и Accept Jobs (Открыть прием заданий), соответственно, и по своему действию эти ссылки становятся эквивалентны командам *enable* и *accept*.

Щелкнув на ссылке Jobs (Задания) в верхней части страницы, можно просмотреть список заданий, находящихся в очереди. После щелчка на этой ссылке будет открыта страница, содержащая информацию, аналогичную той, что выводится командой *lpstat*. Этот список включает в себя колонку Control (Управление), где можно увидеть ссылки Hold Job (Задержать задание) и Cancel Job (Отменить задание) для каждого задания. Щелчком на этих ссылках можно, соответственно, временно приостановить исполнение задания или удалить выбранное задание полностью.

Поддержка совместимости с LPD

CUPS разрабатывалась как замена системы печати LPD и имеет в своем арсенале практически те же команды, что были в LPD. (Они были описаны в разделе «Печать» ранее в этой главе.) Ключом к совместимости с LPD является обеспечение такой среды окружения, в которой смогут работать программы, изначально разрабатывавшиеся для взаимодействия с LPD, и будет возможным прием заданий печати от удаленных клиентов LPD.

Поддержка устаревшего файла /etc/printcap

Многие программы, ориентированные на работу с LPD, просматривают содержимое файла */etc/printcap*, чтобы определить, какие очереди печати имеются в системе. При использовании LPD в этом файле находятся определения всех имеющихся очередей печати, поэтому данный файл имеет большое значение в LPD. Учитывая это обстоятельство, CUPS пытается поддерживать файл */etc/printcap* в минимальном объеме, необходимом для прикладных программ. (Программы, ориентированные на работу с CUPS, получают списки очередей печати иным образом.)

Полный файл */etc/printcap* системы LPD содержит множество типов полей, и любое описание принтера наверняка будет состоять из полудюжины таких полей. При работе с системой CUPS оказалось, что вполне достаточно упрощенного варианта файла, в котором значительную часть сведений можно опустить. Обычно файл выглядит следующим образом:

```
hp4000|Hewlett-Packard HP LaserJet 4000 Series:rm=nessus:rp=hp4000:
epson|Epson RX500 1440x720 dpi:rm=nessus:rp=epson:
epson_360|Epson RX500, 360dpi economy:rm=nessus:rp=epson_360:
```

В этих трех строках описываются три очереди: `hp4000`, `epson` и `epson_360`. В системе LPD каждая очередь печати может иметь несколько имен, отделяемых одно от другого символом `|`. Как правило, определение очереди начинается с самого короткого имени, за которым следуют остальные, более длинные имена. В CUPS с учетом этой особенности короткое имя создается из имени очереди, а более длинное имя – из строки описания принтера. Все остальные поля в каждой строке отделены друг от друга символом двоеточия (`:`). В данном примере системой CUPS были созданы записи, содержащие параметры `rm=` и `rp=`, с помощью которых в LPD определяются имя удаленного сервера и имя удаленной очереди печати для сетевых принтеров. Здесь эти параметры идентифицируют компьютер, где находится файл, и имя очереди печати. Ни CUPS, ни большинство программ, ориентированных на работу с LPD, не требуют наличия этой информации, но она поможет сохранить функциональность некоторых программ, которые вполне удовлетворит этот минимальный объем информации.

Обычно администраторам не приходится задумываться о поддержке файла */etc/printcap*, потому что CUPS делает это автоматически. Тем не менее иногда приходится сталкиваться с необходимостью создавать фиктивный файл */etc/printcap* вручную. Данный файл должен содержать как минимум имена всех очередей и двоеточия. Дополнительные параметры `rm=` и `rp=`, как в предыдущем примере, могут помочь некоторым программам.

Прием заданий печати от систем LPD

Как отмечалось ранее, для организации взаимодействия между системами печати CUPS используется протокол IPP. Системы LPD используют более старый протокол LPD. Таким образом, если в сети имеются системы CUPS и LPD либо системы CUPS и какие-нибудь другие системы печати, которые тем не менее понимают протокол LPD, можно будет включить поддержку LPD в CUPS. Наличие такой поддержки позволит системе CUPS принимать задания печати, отправляемые по протоколу LPD. Данная поддержка не требуется в сетях, где присутствуют только системы CUPS, и она выключена по умолчанию во всех основных дистрибутивах Linux.

Чтобы обеспечить поддержку LPD, достаточно всего лишь разрешить принимать задания печати от клиентов LPD. Если необходимо, чтобы демон CUPS отправлял задания в удаленную систему LPD, включать такую поддержку совсем не обязательно. В этом случае необходимо лишь определить сервер LPD как сетевой сервер печати, о чем уже говорилось в разделе «Создание описания принтера».

Поддержка LPD в CUPS обеспечивается небольшим демоном с говорящим названием *cups-lpd*. Этот сервер разрабатывался так, чтобы работать под управлением суперсервера, такого как *inetd* или *xinetd*. В дистрибутивах, где используется *xinetd*, таких как Fedora, Red Hat и SUSE, в каталоге */etc/xinetd.d* можно найти файл с настройками для *xinetd*, который называется *cups-lpd*. Найдите в этом файле строку `disable = yes` и замените ее строкой `disable = no`. Затем нужно перезапустить сервер *xinetd* или послать ему сигнал, чтобы он перечитал файлы с настройками, после чего должен запуститься сервер *cups-lpd*. Этот сервер будет при-

нимать задания печати по протоколу LPD точно так же, как BSD LPD или LPRng, но будет перенаправлять задания в локальную очередь CUPS с тем же именем.

Если в системе используется сервер *inetd*, запись о сервере *cups-lpd* необходимо добавить в файл */etc/inetd.conf*:

```
printer stream tcp nowait lp /usr/lib/cups/daemon/cups-lpd cups-lpd
```

В некоторых системах может потребоваться изменить эти настройки, например, когда появится необходимость вызывать *cups-lpd* не напрямую, а с помощью обертки TCP (TCP Wrappers – `tcpd`). Как и в случае с *xinetd*, после изменения файлов с настройками следует перезапустить *inetd* или послать ему сигнал, чтобы он перечитал их и запустил сервер *cups-lpd*.

Устранение неполадок в системе печати

Если вы неукоснительно следовали инструкциям в этой главе, то при наличии определенной доли везения вам удастся запустить принтер практически без каких-либо осложнений. Но, к сожалению, тщательная настройка CUPS не всегда дает запланированные результаты, а поиск неисправностей в системе печати может оказаться делом сложным, сокрушающим веру в свои собственные силы. Мы не можем описать вам абсолютно безошибочный способ настройки принтеров, но в наших силах дать некоторые рекомендации, которые могут помочь найти источник проблемы и решить ее.

- Если это возможно, проверьте работоспособность принтера с другой операционной системой на этом же компьютере. Для такой проверки подойдет даже загрузочная дискета FreeDOS (*http://www.freedos.org*), по крайней мере, для принтеров с параллельным и последовательным интерфейсами. Если принтер работает с другой операционной системой, следовательно, проблема не связана с исправностью аппаратной части, в противном случае это предполагает (но не доказывает) неисправность аппаратуры.
- Проверьте еще раз все соединения от кабеля питания и до кабеля соединения с компьютером. Не соединенные или поврежденные кабели могут стать причиной многочасовых бесплодных поисков. Если на корпусе принтера есть выключатель питания, проверьте, включен ли он. (Известны самые невероятные случаи, когда главной проблемой был просто не включенный выключатель!) Кроме того, проверьте наличие бумаги во всех подающих лотках принтера.
- Внимательно осмотрите переднюю или верхнюю панель принтера на предмет наличия сигналов об ошибках. В некоторых принтерах имеются небольшие жидкокристаллические дисплеи, где могут появляться короткие сообщения, например `out of paper` (нет бумаги) или `paper jam` (заедание бумаги). В других принтерах имеются светодиодные индикаторы, которые начинают мигать при появлении какой-либо неисправности или ошибки. Просмотрите руководство по эксплуатации принтера, где должно быть написано, как следует интерпретировать эти сигналы.
- Если у вас принтер с параллельным или с последовательным интерфейсом, попробуйте послать задание на печать, используя «сырой» файл устройства, как было описано ранее, в разделе «Проверка совместимости принтера». Если это не сработает, но принтер физически исправен, стоит проверить нали-

чие драйвера устройства в ядре. Загрузка драйвера с помощью *modprobe*, возможно, решит эту проблему. Другая возможная причина состоит в том, что используется не тот файл устройства, поэтому можно попробовать использовать другие файлы устройств. Если светодиодные индикаторы принтера мигают при попытке напечатать что-нибудь, это может свидетельствовать о том, что файл поступает в принтер, но имеет формат, который принтер не может понять, отсюда можно сделать заключение (но не окончательное), что основные функции оборудованием выполняются.

- Если речь идет о принтере USB, с помощью команды *lsusb* убедитесь, что ядро верно идентифицирует принтер, и внимательно просмотрите содержимое файла */proc/bus/usb/devices*, о чем уже говорилось в разделе «Проверка совместимости принтера». Если принтер не опознается ядром, возможно, не были загружены драйверы ядра или принтер просто не поддерживается Linux.
- В случае с принтером, имеющим сетевой интерфейс, попробуйте проверить соединение с ним с помощью утилиты *ping*. Если принтер не отвечает, обратитесь к руководству по эксплуатации принтера, чтобы узнать, как активизировать его сетевой интерфейс. Если IP-адрес принтеру назначается с помощью DHCP, внимательно проверьте параметры настройки сервера DHCP и файлы журналов, чтобы выяснить, почему принтер не воспринимает назначаемый ему IP-адрес.
- Если принтер имеет несколько интерфейсов подключения (например, параллельные порты и порты USB), попробуйте использовать интерфейс, который ранее не использовался. Если принтер заработает (после того, как будет произведена перенастройка параметров в Linux), это может означать, что не работали драйверы интерфейса, к которому принтер подключался первоначально, или мог быть поврежден порт интерфейса в принтере или в компьютере, или имеются дефекты в соединительном кабеле. В этом случае можно продолжить поиск проблемы или использовать работоспособный интерфейс.
- Если попытки печати через файл устройства увенчались успехом либо сетевой принтер или принтер USB определяются и откликаются на основные команды, следует перейти к тщательной проверке настроек CUPS.
- Находясь на главной странице веб-интерфейса CUPS (см. рис. 14.3), проверьте еще раз описание принтера (или введите команду *lpc status* в командной строке). Если из описания следует, что принтер остановлен или заблокирован, щелкните на соответствующей ссылке или воспользуйтесь командами *enable* и *accept*, чтобы запустить очередь в работу.
- Если описание принтера включает в себя фразу `waiting for job to complete` (ожидание завершения выполнения задания), значит, что-то мешает CUPS очистить очередь печати. Это может быть связано с отсутствием соединения с принтером или с программной ошибкой. Иногда в таких ситуациях проблема решается простым перезапуском CUPS.
- Если принтер печатает, но на бумаге воспроизводится всякий мусор, это может означать, что при настройке очереди была выбрана не та модель принтера. (Иногда такое может происходить, если задание было прервано и принтер был включен до того, как очередь была очищена, но это не означает, что ошибка связана с драйвером.) Если выводимый текст похож на программу PostScript, возможно, был выбран вариант PostScript для принтера, не под-

держивающего PostScript, отсюда следует, что выбор в настройках модели без поддержки PostScript может решить эту проблему. Если текст представляет собой практически случайный набор символов, вероятнее всего, был выбран полностью несовместимый принтер без поддержки PostScript.

- Серьезной проблемой может стать низкая скорость печати. Нередко скорость удается поднять (особенно для струйных принтеров), снизив разрешающую способность. Это ухудшит качество печати, поэтому вам придется искать золотую середину. Если принтер USB 2.0 печатает слишком медленно, убедитесь, что он подключен к порту компьютера USB 2.0, и что были загружены драйверы USB 2.0 (EHCI). В противном случае загрузите соответствующие драйверы и в случае необходимости приобретите дополнительную плату USB 2.0.
- Если качество печати не соответствует вашим ожиданиям, попробуйте изменить параметры настройки, описанные в разделе «Дополнительная настройка описаний принтеров». Возможно, проблему удастся снять, увеличив разрешающую способность, настроив цветопередачу, выбрав другой уровень яркости или как-то иначе изменив способ, которым Ghostscript и принтер обрабатывают документы. Изменения некоторых из этих параметров могут привести к снижению скорости печати.

В общем и целом, поиск неисправностей в системе печати можно считать и величайшим искусством, и наукой. Многое может пойти не так, как надо, и трудно заранее предсказать, что может случиться с наибольшей вероятностью. Данные рекомендации по крайней мере помогут вам сузить список потенциальных причин и укажут направление, в котором следует двигаться, чтобы обнаружить и ликвидировать возникшую проблему.

За кулисами: файлы и каталоги CUPS

В предыдущем обсуждении практически ничего не говорилось о файлах и каталогах системы печати CUPS. Это произошло по той простой причине, что редактирование файлов настроек CUPS вручную, за исключением */etc/cups/cupsd.conf*, требуется довольно редко. Зачастую CUPS реагирует на такие вмешательства достаточно неадекватно, поэтому желательно все изменения в настройках CUPS делать с использованием веб-интерфейса и команд системы печати.

После всего сказанного, возможно, вы захотите узнать подробнее об этих файлах и каталогах. Файлы с настройками CUPS находятся в каталоге */etc/cups*. Файл *cupsd.conf*, как уже отмечалось, содержит параметры настроек глобального уровня, и этот файл не просто можно, но и нужно редактировать. Из других важных файлов в этом каталоге можно отметить *printers.conf* (который содержит описания локальных принтеров) и *lpoptions* (идентифицирует принтер по умолчанию). Подкаталог *ppd* содержит файлы PPD для локальных принтеров (они копируются из других мест, о чем вскоре будет упомянуто).

Большой объем данных, принадлежащих CUPS, хранится в дереве каталогов, ниже */usr/share/cups*. Отдельный интерес может представлять подкаталог */usr/share/cups/model*, где хранятся файлы PPD (чаще всего в подкаталогах с именами, соответствующими названиям фирм-изготовителей принтеров). При установке пакетов Foomatic и GIMP Print файлы PPD могут копироваться в этот каталог. Если это не так и CUPS не видит эти описания принтеров, попробуйте соз-

дать символические ссылки в этом подкаталоге на фактическое размещение файлов PPD из пакета драйверов принтеров. Это позволит системе CUPS определить местонахождение файлов PPD и установить принтеры.

В процессе своей работы CUPS использует подкаталог */var/spool/cups* для хранения описаний заданий печати и самих файлов заданий. Владельцем этого подкаталога является пользователь *root* и группа *lp*, права доступа к подкаталогу определены как 0710 (`rwx--x---`). Кроме того, CUPS использует каталог */var/spool/cups/tmp* как справочник, который также принадлежит пользователю *root* и группе *lp*, с правами доступа 1710 (`rwx--x--T`). (Эти владельцы и права доступа типичны, но в некоторых дистрибутивах они могут быть иными.) В отличие от LPD, CUPS не использует отдельные подкаталоги для каждого принтера, все задания для всех принтеров помещаются в один и тот же каталог.

15

Совместное использование файлов

Данная глава представляет собой краткое руководство по двум основным способам совместного использования ресурсов разными системами. Первым будет описан пакет Samba, в котором используются сетевые протоколы Microsoft Windows, позволяющие пользователям одной системы читать и записывать файлы в другую систему, а также посылать задания принтерам на удаленных системах. Основное преимущество пакета Samba состоит в том, что он позволяет без особых затруднений объединить Linux и UNIX с системами Microsoft, как клиенты, так и серверы. Сетевые протоколы Microsoft Windows могут использоваться для организации совместного доступа к файлам и между системами Linux, хотя в такой ситуации предпочтительнее использовать протокол NFS.

Затем будут представлены протоколы NFS и NIS, разработанные Sun Microsystems и используемые системами UNIX уже в течение многих десятилетий. Протокол NFS (Network File System – сетевая файловая система) позволяет организовать совместный доступ к файлам для систем Linux и UNIX способом, аналогичным используемому в Samba. Протокол NIS (Network Information System – сетевая информационная система) позволяет сохранять пользовательские данные в одном месте и предоставлять доступ к ним из разных систем, благодаря чему можно избежать необходимости обновления всех систем, когда изменяются имя пользователя или пароль. Хотя NIS не относится к средствам организации совместного доступа к файлам и принтерам, тем не менее данный протокол будет описан в этой главе, потому что в нем используются некоторые компоненты родственного ему протокола NFS, а также потому, что с помощью NIS существенно упрощается администрирование NFS, так как NIS позволяет каждому пользователю иметь одну и ту же учетную запись на всех системах.

Протоколы NFS и NIS очень удобны в сетях, где находятся только системы Linux и UNIX. В свое время были созданы версии этих протоколов для систем Microsoft, но они не отличались устойчивостью и не достигли популярности.

В Microsoft были созданы реализации клиента и сервера NFS для систем Windows, которые, впрочем, не получили широкого распространения несмотря на то, что они распространяются бесплатно. В состав пакета Microsoft Windows Services for UNIX (SFU) вошли сервер NIS и более 300 утилит UNIX для использования в операционной системе Windows. Но несмотря на наличие этих бесплатных реа-

лизаций, для организации надежного взаимодействия сетевых клиентов Windows с системами Linux предпочтительнее использовать протокол Samba.

Помимо сетевых протоколов MS Windows и NFS существуют и другие протоколы совместного доступа к файлам и принтерам. В частности, Linux поддерживает возможность организации совместного доступа к файлам и принтерам аналогично системе NetWare с использованием протокола IPX или Macintosh (протокол AppleTalk). Существует возможность совместного доступа к файлам по таким протоколам, как FISH (File Sharing over SSH – совместный доступ к файлам через SSH), а также имеется поддержка служб доступа к файлам на основе WebDAV. В данной главе мы не будем касаться этих протоколов.

Совместный доступ к файлам вместе с Windows (Samba)

Революция программного обеспечения с открытыми исходными текстами еще не закончена, и в наши дни используется еще очень много настольных компьютеров и серверов, работающих под управлением Windows. Некоторые полагают, что в скором будущем весь мир будет использовать только Linux, действительность же говорит нам обратное: настольные системы Windows будут существовать еще достаточно долго. Таким образом, возможность обмениваться файлами между Windows и Linux имеет очень большое значение. Не менее важна и возможность совместного использования принтеров.

Samba – это очень гибкий и легко масштабируемый набор приложений, позволяющий пользователям Linux читать и записывать файлы, расположенные на рабочих станциях Windows, и наоборот. Такую возможность можно использовать, например, с целью организации доступа к файлам в системе Linux для единственного клиента Windows (например, для Windows, запущенной в виртуальной аппаратной среде на ноутбуке с операционной системой Linux). Но Samba может с неменьшим успехом использоваться для создания надежного и высокоэффективного файлового сервера и сервера печати в сети, имеющей тысячи клиентов Windows. Если предполагается крупномасштабное использование Samba, тогда, вероятно, вам следует потратить определенное время на чтение обширной документации Samba, которую можно найти на сайте *http://www.samba.org/samba/docs*, или хорошую книгу, например «Using Samba» (O'Reilly), которая входит в состав дистрибутива Samba.

В этом разделе описываются ключевые аспекты, которые необходимо знать о возможности совместного использования файлов и принтеров операционными системами Windows и Linux. Для начала вашему вниманию будет представлен краткий обзор основных принципов работы с сетями в операционной системе Windows, чтобы помочь избежать расстройств и мучений, которые часто испытывают те, кто впервые пытаются пересечь горный хребет, отделяющий мир Windows от мира UNIX. Затем коротко будет рассказано об инструментальных средствах, имеющихся в стране Linux, которые помогут пользователю Linux получать доступ к файлам и принтерам, живущим в стране Windows. Тема предоставления пользователям Windows доступа к файлам и принтерам, размещенным в системе Linux, будет рассмотрена последней, и вовсе не потому, что эта тема менее важна, а потому, что диапазон возможностей в этом направлении гораздо шире.

Протоколы и другие особенности, имеющие отношение к Windows

Пользователям Linux хорошо известно: все, что им необходимо знать для организации доступа к удаленным системам Linux, – это IP-адрес. По сути, IP-адрес вкупе с системой доменных имен (DNS) представляют собой совершенный механизм обеспечения взаимодействия между любыми системами Linux. Поэтому мы можем сказать, допустив некоторое поэтическое отступление, что пространство имен Linux есть DNS. В пространстве имен мира TCP/IP имеются некоторые ограничения на максимально допустимую длину имени хоста или имени, которое может быть помещено в базу данных DNS. Но порог человеческой лени обычно находится ниже максимального количества символов, допустимого в имени хоста.

Жизнь в стране Windows совсем не так безоблачна, и тому есть веские причины. Мир сетей Windows находится в совершенно ином пространстве имен, что явилось следствием попытки решить проблему совместного использования файлов без непосредственного участия протоколов TCP/IP. Протоколы TCP/IP были привлечены к работе позднее, но с самого начала стек протоколов TCP/IP отсутствовал в Windows. Родным сетевым протоколом для этой операционной системы был NetBEUI (Network Basic Extended User Interface – основной расширенный сетевой пользовательский интерфейс). Для склонных к соблюдению технических точностей такое название может служить примером ошибочного употребления, потому что протокол фактически состоит из протокола SMB (Server Message Block – блок серверных сообщений), транспортируемого по протоколу NetBIOS и обернутого протоколом адресации LLC (Logical Link Control – протокол управления логическим соединением). Получившийся в результате протокол оказался не маршрутизируемым и довольно неэффективным. Прежнее название протокола – SMB – дало начало названию программного проекта Samba, созданного разработчиком Эндрю Трайджеллом (Andrew Tridgell), когда он решил эмулировать протокол совместного доступа к файлам в Windows.

Некоторое время спустя, приблизительно в 1996 году, протокол SMB был переименован в протокол CIFS (Common Internet File System – общая файловая система Интернета). Изначально протокол CIFS по сути был протоколом SMB. При употреблении в широком смысле эти два термина можно считать взаимозаменяемыми. Протокол SMB/CIFS поддерживает следующие функциональные возможности:

- Доступ к файлам.
- Блокировка файлов и записей.
- Возможность получения извещений при изменении файла.
- Возможность договариваться об использовании определенной версии протокола.
- Расширенный набор атрибутов файлов и каталогов.
- Организация распределенных виртуальных файловых систем.
- Независимое разрешение имен.
- Использование кодировки Unicode для имен файлов и каталогов.

Рассмотрение всех этих особенностей далеко выходит за рамки данной книги, поэтому заметим лишь, что при правильных настройках протоколы работают достаточно хорошо даже в масштабах крупных предприятий.

Протокол NetBIOS фактически представляет собой прикладной программный интерфейс (API), позволяющий при минимальном объеме кодирования выполнять операции SMB/CIFS передачи данных по транспортному протоколу некоторого типа. Протокол NetBEUI, также известный как протокол NetBIOS Frame (NBF), для адресации использует протокол LLC. Этот протокол появился в 1980-х годах и сначала использовался фирмой IBM в составе предлагаемого ею продукта PC-LAN. Реализация NetBIOS через TCP/IP появилась позднее и была описана различными стандартами. Протокол NetBIOS допускает возможность инкапсуляции во многие другие протоколы, из которых самым известным является протокол системы Netware – IPX/SPX.

Протокол NetBIOS (точнее, SMB) обладает своим собственным пространством имен. В отличие от пространства имен TCP/IP, все имена NetBIOS могут иметь длину до 16 символов, однако использовать в этих именах символы, отличные от алфавитно-цифровых, было бы весьма неосмотрительно. Попытка использовать цифру в качестве первого символа имени определенно потерпит неудачу, поскольку реализация NetBIOS через TCP/IP будет интерпретировать такое имя как IP-адрес. Шестнадцатый символ в имени NetBIOS – это символ типа имени, который используется серверами и клиентами для обнаружения определенных сетевых служб, таких как служба сетевой регистрации в системе.

Пространство имен NetBIOS включает в себя такое понятие, как *рабочая группа* (*workgroup*). Компьютеры, имеющие одинаковое имя рабочей группы, принадлежат к одной рабочей группе. В сервере локальной сети IBM (IBM Lan Server) и диспетчере локальной сети Microsoft (Microsoft LAN Manager, входивший в состав Windows NT4) использовался термин *домен*, что указывало на использование некоторой технологии аутентификации, но на самом низком уровне домен был полностью идентичен рабочей группе.

В сетях, основанных на NetBIOS, чрезвычайно важно было настроить каждую машину на использование одних и тех же сетевых протоколов, а все протоколы должны были настраиваться идентичным образом. Отклонения не допускались, в противном случае любая попытка сделать что-то по-разному приводила к отказам сетевых взаимодействий.

Протокол NetBIOS через TCP/IP (NBT, или NetBT) использует два основных протокола и два номера порта для выполнения основных операций: порт TCP с номером 139 (порт службы сеанса NetBIOS – NetBIOS Session Service) и порт UDP с номером 137 (порт сервера имен NetBIOS – NetBIOS Name Server). Порт UDP 137 используется для разрешения имен на основе широковещательных посылок с помощью метода, известного как широковещательная рассылка извещений. Такая сетевая активность систем может порождать существенный объем трафика.

Лучший способ уменьшить объем широковещательного трафика по протоколу UDP заключается в использовании сервера имен NetBIOS. В Microsoft эта разновидность сервера получила название WINS (Windows Internet Naming Service). Сервер WINS для NetBT – это то же самое, что DNS для TCP/IP. Клиенты регист-

рируют свои имена NetBIOS в сервере WINS при запуске. Если все машины сети настроены на выполнение запросов к серверу WINS, сетевые взаимодействия между системами Windows обычно протекают без особых осложнений. Сервер WINS предоставил практичную и эффективную возможность преобразования имен NetBIOS в соответствующие им IP-адреса.

С появлением Windows 2000 компания Microsoft ввела новую технологию, получившую название Active Directory, в которой для преобразования имен компьютеров в IP-адреса использовалась служба DNS. В сетях, где в качестве клиентов и серверов использовались только системы Windows 2000 (или более поздние версии), наряду с технологией Active Directory компания Microsoft обеспечила возможность полного отказа от использования протокола NetBIOS. Его место заняла новая технология, в которой используется протокол SMB через TCP/IP. Эта технология известна как NetBIOS-less TCP/IP. При отсутствии широковещательного способа разрешения имен на основе протокола UDP и отказа от WINS, которые входят в состав набора протоколов NetBT, протокол NetBIOS-less TCP/IP основан на разрешении имен с помощью службы DNS и использовании системы безопасности Kerberos в паре со службой Active Directory. Active Directory – это более или менее совместимая реализация стандарта облегченного протокола службы каталогов (Lightweight Directory Access Protocol, LDAP), для которого имеется превосходная свободно распространяемая реализация под названием OpenLDAP (упоминавшаяся в главе 8), которая позволяет эмулировать наиболее важные службы, предлагаемые Active Directory, с помощью операционной системы Linux.

Чтобы иметь возможность использовать Samba без поддержки NetBIOS, система должна быть одним из серверов-членов домена Active Directory. Не следует отключать поддержку NetBIOS, если система не настроена на работу с Active Directory.

Протокол Samba версии 2 может использоваться только совместно с протоколом NetBT. В версии 3 Samba в состоянии обеспечить интеграцию с сетями Windows, обладающими поддержкой Active Directory и работающими на основе протокола NetBIOS-less TCP/IP. При работе в таком окружении протокол должен быть настроен на использование порта TCP с номером 445, который используется сетевой подсистемой Windows при отсутствии поддержки NetBIOS. Кроме того, сетевая подсистема Microsoft Windows использует порт TCP с номером 135 для организации взаимодействий по протоколу DCE RPC (протокол вызова удаленных процедур). Обсуждение этих протоколов не укладывается в рамки данной книги. Основное внимание будет сконцентрировано на использовании протокола Samba с NetBT.

Третья версия реализации протокола Samba появилась в сентябре 2003 года, после более чем двухгодичной разработки. В этой версии гораздо полнее была реализована поддержка сетевых протоколов Windows 200x, добавлена поддержка Unicode, появилась возможность хранения паролей различными способами (включая LDAP) и присоединения к доменам Active Directory Windows 200x с использованием протокола безопасности Kerberos. Данная версия остается текущей стабильной версией и по-прежнему активно развивается с намерением сохранить ее текущей и в 2007 году. Группа разработчиков Samba предполагала

выпустить четвертую бета-версию Samba к концу 2005 года, после почти трех лет разработки. Четвертая версия Samba переписана с самого начала. Она обладает обширной поддержкой технологии Active Directory с целью обеспечить возможность управления доменом Active Directory. Ожидалось, что достаточного уровня стабильности Samba 4 достигнет к середине 2006 года, что позволило бы любителям новизны начать переход на новую версию.

Если только возможно, пакет Samba должен использоваться совместно с существующим сервером Microsoft WINS или сам выступать в роли сервера WINS, чтобы обеспечить разрешение имен NetBIOS. Не забывайте: цена за неиспользование WINS – увеличение объема широковещательного трафика UDP и снижение доступности сетевых служб.

Этот раздел мы начнем с примера, в котором рассмотрим возможность организации доступа к файлам, находящимся на сервере Windows, со стороны Linux. Изначально предполагается, что соединение по протоколу TCP/IP между Linux и компьютерами Windows уже установлено, и что в системе Windows настроен разделяемый каталог. Детальные инструкции по организации совместного доступа к файлам в операционных системах Windows 95/98/Me и Windows NT/2000/XP можно найти в книге «Using Samba» (O'Reilly).

Начнем с того, что обе операционные системы – и Windows, и Linux – должны быть корректно настроены для обеспечения возможности взаимодействия по протоколу TCP/IP. Это означает следующее:

- Каждой системе должен быть присвоен корректный IP-адрес.
- В обеих системах должны быть настроены правильные сетевые маски.
- Системы должны обращаться к одному и тому же шлюзу (если одна из локальных сетей содержит маршрутизаторы для выхода в другие сегменты сети).
- В каждой системе имеется корректный файл */etc/hosts* и должным образом настроена работа со службой DNS, если таковая используется.

Имя компьютера с Windows и название рабочей группы должны состоять только из алфавитно-цифровых символов. Если предполагается выполнять разрешение имен на основе файла */etc/hosts*, аналогичный ему файл в системе Windows должен носить имя *hosts*, без расширения. В системах Windows 95/98/Me файл *hosts* должен находиться в каталоге *C:\Windows\System*. В системах Windows NT/2000/XP – в каталоге *C:\Winnt\System32\drivers\etc*.

Образец файла *hosts* в Windows NT/2000/XP имеет расширение *sam*. Не используйте этот файл в качестве рабочего, поскольку файл с расширением не будет использоваться сетевой подсистемой.[1]

В остальной части этой главы будет использоваться термин *«имя SMB»* для обозначения имени NetBIOS системы, обладающей поддержкой SMB (известно также как *«имя машины»*). Под термином «*рабочая группа*» (*workgroup*) будут подразумеваться и имя рабочей группы, и имя домена машины с поддержкой SMB.

---

[1] Используя этот образец, нужно создать свой файл *hosts* по образу и подобию *hosts.sam*, вписав туда собственные имена хостов и их IP-адреса. – *Примеч. науч. ред.*

Обратите внимание: с точки зрения типичных сетевых операций, таких как обзор доменов или рабочих групп и обзор разделяемых ресурсов на машинах, понятия *«имя рабочей группы»* и *«имя домена»* взаимозаменяемы, поэтому мы будем использовать термин *«рабочая группа»*.

В данном примере под машиной с Windows будет подразумеваться компьютер, работающий под управлением операционной системы Windows XP Home и имеющий имя `EMACHO` и IP-адрес 192.168.1.250. Рабочая группа будет называться `MIDEARTH`. Система Linux имеет имя хоста `loudbell`, IP-адрес 192.168.1.4 и находится в домене `goodoil.biz`.

Подготовка системы Linux и установка пакета Samba

Службы, обсуждаемые в этой главе, требуют наличия модулей ядра и инструментальных средств, которые изначально могут отсутствовать в вашей системе Linux. Многие коммерческие версии Linux (Novell SUSE Linux и Red Hat Linux) поставляются со всем необходимым. Если вы используете систему Linux, собранную самостоятельно, вам, возможно, придется пересобрать ядро. Шаги, описанные в этом разделе, должны в этом помочь. И, конечно же, потребуется пакет Samba версии 3.0.x.

Для начала необходимо заглянуть в настройки ядра и убедиться, что в нем включена поддержка всего необходимого.

Прежде всего, в ядре должна быть включена поддержка `smbfs` и `cifsfs`. Если в системе используется старая версия ядра (до версии 2.6.x), поддержка `cifsfs` в нем может отсутствовать. Однако существуют драйверы поддержки `cifsfs` для устаревших версий ядра, которые можно будет установить. За дополнительной информацией, касающейся поддержки `cifsfs`, обращайтесь на сайт проекта CIFS: *http://linux-cifs.samba.org/*. Прежде чем добавлять этот модуль в дерево каталогов с исходными текстами ядра, внимательно ознакомьтесь с инструкциями на этом сайте.

Модули ядра `smbfs` и `cifsfs` не являются частью Samba. Каждый из них представляет собой отдельный проект по разработке драйвера ядра. Оба проекта необходимы для обеспечения работоспособности вспомогательных инструментальных средств, таких как `smbmount`, `smbumount`, `mount.smbfs` и `mount.cifs`, которые распространяются в составе дистрибутива Samba и требуют наличия этих драйверов.

Исходные тексты ядра версии 2.6.x включают в себя модуль `cifsfs`. Чтобы узнать, поддерживает ли текущее рабочее ядро необходимые возможности, нужно установить исходные тексты ядра в каталог */usr/src/linux*, а затем выполнить следующие действия:

1. Создать файл с настройками в исходных текстах ядра, чтобы он соответствовал настройкам текущего рабочего ядра:

```
linux:~ # cd /usr/src/linux
linux:~ # make cloneconfig
```

   По окончании клонирования настроек содержимое файла с настройками будет выведено на экран. Пока не обращайте внимания на мелькающие строки,

потому что вся информация будет также сохранена в файле *.config*, который будет исследоваться в следующем шаге.

2. Чтобы определить наличие в ядре поддержки `smbfs`, нужно выполнить следующую команду:

```
linux:~ # grep CONFIG_SMB_FS .config
CONFIG_SMB_FS=m
```

В данном случае результат выполнения команды говорит, что поддержка `smbfs` в ядре имеется, и что она выполнена в виде загружаемого модуля ядра. Значение *y* означало бы, что данный модуль встроен прямо в ядро, что также приемлемо, а значение *n* означало бы отсутствие поддержки.

В случае отсутствия поддержки `smbfs` следует воспользоваться утилитой настройки ядра, которая описывается в разделе «Настройка ядра: make config» главы 18, чтобы включить ее.

3. Теперь следует определить наличие поддержки `cifsfs` в текущем ядре:

```
linux:~ # grep CONFIG_CIFS .config
CONFIG_CIFS=m
```

Такой результат означает, что поддержка `cifsfs` в ядре имеется. Если будет получено значение *n*, следует включить данную поддержку с помощью утилиты настройки.

4. Если на одном из предыдущих шагов пришлось воспользоваться утилитой настройки ядра, необходимо будет пересобрать и установить ядро.

После перезагрузки системы новое ядро будет готово к выполнению следующих шагов, описываемых в этой главе. На следующем этапе необходимо получить свежую версию Samba.

Скомпилированные пакеты Samba уже входят в состав большинства дистрибутивов Linux и UNIX. Кроме того, пакеты Samba можно найти на домашней странице проекта Samba: *http://samba.org*.

За подробным описанием порядка установки пакетов в конкретной системе следует обращаться к справочному руководству по этой операционной системе. В очень редких случаях может потребоваться скомпилировать пакет Samba. Информацию, которая поможет облегчить процесс компиляции и установки, можно найти в документе «Samba-3-HOWTO» по адресу: *http://www.samba.org/samba/docs/Samba3-HOWTO.pdf*.

Если будет принято решение собрать и установить Samba вручную, предварительно необходимо удалить все пакеты Samba, которые поставлялись в составе дистрибутива и, возможно, уже были установлены в системе. Если этого не сделать, тогда на исполнение могут быть запущены файлы, оставшиеся от прежнего пакета, что, в свою очередь, может вызвать хаос и беспорядок и доставить немало неприятных минут.

Прежде чем запустить компиляцию Samba, необходимо выполнить команду *configure* с параметром *--with-smbmount*. Следующие команды выполнят сборку и установку нового пакета Samba:

```
linux:~ # make all libsmbclient wins everything
linux:~ # make install
linux:~ # make install-man
```

Когда процесс сборки и установки завершится, следует запустить следующие команды, которые обеспечат сборку и установку исполняемого файла `mount.cifs`:

```
linux:~ # export CFLAGS="-Wall -O -D_GNU_SOURCE -D_LARGEFILE64_SOURCE"
linux:~ # gcc client/mount.cifs.c -o client/mount.cifs
linux:~ # install -m755 -o root -g root client/mount.cifs /sbin/mount.cifs
```

Теперь систему можно считать готовой к дальнейшей настройке, поэтому перейдем к упражнениям в организации совместного доступа к файлам из *другого мира*.

Доступ к удаленным файлам и принтерам Windows

В ближайшее время мы подключимся к разделяемому ресурсу системы Windows. Далее будем исходить из предположения, что Windows имеет статический IP-адрес и служба DNS не используется. Возможность разрешения имен имеет очень большое значение при организации сетевых взаимодействий, особенно это относится к клиентам Windows, поэтому начнем с подготовки файла */etc/hosts*, в который добавим следующую строку:

```
192.168.1.250 emacho
```

Разумеется, здесь же должна быть строка с IP-адресом системы Linux.

Теперь проверим возможность разрешения имен с помощью файла */etc/hosts*:

```
linux:~ # ping emacho
PING emacho (192.168.1.250) 56(84) bytes of data.
64 bytes from emacho (192.168.1.250): icmp_seq=1 ttl=128 time=2.41 ms
64 bytes from emacho (192.168.1.250): icmp_seq=2 ttl=128 time=2.16 ms
64 bytes from emacho (192.168.1.250): icmp_seq=3 ttl=128 time=2.16 ms
64 bytes from emacho (192.168.1.250): icmp_seq=4 ttl=128 time=2.02 ms
64 bytes from emacho (192.168.1.250): icmp_seq=5 ttl=128 time=2.01 ms
64 bytes from emacho (192.168.1.250): icmp_seq=6 ttl=128 time=3.90 ms

--- emacho ping statistics ---
6 packets transmitted, 6 received, 0% packet loss, time 5004ms
rtt min/avg/max/mdev = 2.015/2.447/3.905/0.667 ms
```

Замечательно, все работает. Вот теперь мы действительно готовы к совместному использованию файлов.

Организация доступа к Windows с помощью FTP-подобного клиента smbclient

В первую очередь, необходимо убедиться, что система Linux имеет возможность взаимодействовать с системой Windows с использованием Samba. Самый простой способ заключается в использовании клиента Samba – команды *smbclient*, с помощью которого можно отправить запрос системе Windows и узнать, какие ресурсы являются доступными.

Попробуем отправить системе Windows анонимный запрос:

```
linux:~ # smbclient -L emacho -U%
```

```
Domain=[MIDEARTH] OS=[Windows 5.1] Server=[Windows 2000 LAN Manager]

        Sharename       Type      Comment
        ---------       ----      -------
Error returning browse list: NT_STATUS_ACCESS_DENIED
Domain=[MIDEARTH] OS=[Windows 5.1] Server=[Windows 2000 LAN Manager]

        Server               Comment
        ---------            -------

        Workgroup            Master
        ---------            -------
```

Не слишком обнадеживающий ответ, не так ли? Попытка потерпела неудачу, о чем свидетельствует сообщение `Error returning browse list: NT_STATUS_ACCESS_DENIED` (Ошибка получения списка: NT_STATUS_ACCESS_DENIED). Эта ошибка обусловлена настройками системы Windows, которая не допускает возможность анонимного доступа. Теперь попробуем повторить попытку, но на этот раз с корректной учетной записью, которая была создана в Windows XP Home.

В нашем примере была создана учетная запись для пользователя *lct* с паролем `2bblue4u`. Что ж, приступим:

```
linux:~ # smbclient -L emacho -Ulct%2bblue4u
Domain=[EMACHO] OS=[Windows 5.1] Server=[Windows 2000 LAN Manager]

        Sharename       Type      Comment
        ---------       ----      -------
        IPC$            IPC       Remote IPC
        SharedDocs      Disk
        print$          Disk      Printer Drivers
        Kyocera         Printer   Kyocera Mita FS-C5016N KX
Domain=[EMACHO] OS=[Windows 5.1] Server=[Windows 2000 LAN Manager]

        Server               Comment
        ---------            -------

        Workgroup            Master
        ---------            -------
```

Получилось! Теперь нам известно, что на удаленной системе имеется разделяемый ресурс с именем `SharedDocs`. В следующем шаге мы попробуем подключиться к этому ресурсу и убедимся, что получили работоспособное соединение Samba.

На этом шаге мы подключимся к разделяемому ресурсу, попробуем получить список файлов, а затем скопируем к себе какой-нибудь файл. Это довольно интересный пример использования утилиты *smbclient*:

```
linux:~ # smbclient //emacho/SharedDocs -Ulct%2bblue4u
Domain=[EMACHO] OS=[Windows 5.1] Server=[Windows 2000 LAN Manager]
smb: \>
```

Снова удача! Пока все идет просто замечательно. Попробуем получить список файлов в каталоге:

```
smb:\ dir
.                                   DR        0 Thu May 19 12:04:47 2005
..                                  DR        0 Thu May 19 12:04:47 2005
```

```
  AOL Downloads                       D        0 Tue Sep 30 18:55:16 2003
  CanoScanCSUv571a                    D        0 Thu May 19 12:06:01 2005
  desktop.ini                       AHS      129 Sun Jul 4 22:12:14 2004
  My Music                            DR       0 Sat Apr 16 22:42:48 2005
  My Pictures                         DR       0 Tue Sep 30 18:36:17 2003
  My Videos                           DR       0 Thu Aug 5 23:37:56 2004

            38146 blocks of size 1048576. 31522 blocks available

  smb: \>
```

Теперь перейдем в каталог *CanoScanCSUv571a*:

```
  smb: \> cd CanoScanCSUv571a
  smb: \CanoScanCSUv571a\>
```

И получим список файлов для этого каталога:

```
  smb: \CanoScanCSUv571a\> dir
  .                                   D        0 Thu May 19 12:06:01 2005
  ..                                  D        0 Thu May 19 12:06:01 2005
  CanoScanCSUv571a.exe                A  3398144 Thu Mar 13 22:40:40 2003
  Deldrv1205.exe                      A    77824 Fri Apr 26 14:51:02 2002
  N122U.cat                           A    13644 Tue May 21 02:44:30 2002
  N122u.inf                           A     6151 Tue Apr 16 22:07:00 2002
  N122UNT.cat                         A    15311 Tue May 21 02:44:32 2002
  N122USG                             D        0 Thu May 19 12:10:40 2005
  USBSCAN.SYS                         A     8944 Fri Jun 12 13:01:02 1998

            38146 blocks of size 1048576. 31522 blocks available

  smb: \CanoScanCSUv571a\>
```

Отлично. Пока все получается. Попробуем загрузить файл к себе. Загрузка и выгрузка файлов при работе с *smbclient* производится точно так же, как при работе с клиентом FTP:

```
  smb: \CanoScanCSUv571a\ >get Deldrv1205.exe
  getting file \CanoScanCSUv571a\Deldrv1205.exe of size 77824
           as Deldrv1205.exe (275.4 kb/s) (average 275.4 kb/s)
```

Все вышло, как и должно было быть. На этом данная демонстрация заканчивается. Вернемся обратно в командную оболочку:

```
  smb: \CanoScanCSUv571a\> quit
  linux:~ #
```

Подведем итоги и коротко опишем, что нам удалось выяснить о нашем окружении:

- Существует возможность взаимодействия по протоколу TCP/IP между системами Linux и Windows.
- Анонимный доступ к системе Windows XP Home невозможен.
- Доступ к системе Windows можно получить с использованием учетной записи, созданной в этой системе.

Утилита *smbclient* обладает очень большой гибкостью. Она используется как часть утилиты *smbprint*, где отвечает за передачу потока данных, отправляемых

в удаленную очередь печати SMB/CIFS, примерно так же, как выполнялась передача файла в примере выше. За дополнительной информацией об утилите *smbclient* обращайтесь к страницам справочного руководства.

Когда проверена и подтверждена возможность взаимодействия по протоколу SMB/CIFS, будет не слишком сложно смонтировать тот же самый ресурс с помощью `smbfs`. Попробуем сделать это в следующем разделе.

Организация доступа к Windows с помощью модуля ядра smbfs

Перед тем как двинуться дальше, коротко рассмотрим, что делает драйвер файловой системы `smbfs`. Этот инструмент имеет ограничения, о которых некоторые пользователи Linux не подозревают.

Драйвер файловой системы `smbfs` по своему поведению напоминает утилиту *smbclient*. Он выполняет аутентифицированное соединение с сервером SMB/CIFS на основе предоставленного имени и пароля и после этого дает разрешение на подключение соединения с сервером SMB/CIFS к точке монтирования в файловой системе Linux. Права владения точкой монтирования устанавливаются в соответствии с идентификаторами (ID) пользователя и группы для пользователя Linux, установившего соединение, а права доступа определяются в соответствии со значением параметра `umask`, имевшимся на момент монтирования.

Фактически управление доступом ко всем файлам и каталогам переходит под контроль средств управления разрешениями файловой системы Linux, а на сервере SMB/CIFS все обращения будут рассматриваться как операции, выполняемые единственным пользователем. Несколько пользователей, одновременно работающих в системе Linux и выполняющих доступ к файлам через точку монтирования, будут рассматриваться системой Windows как единственный пользователь, и потому на стороне Windows все попытки доступа будут обслуживаться единственным процессом.

Драйвер файловой системы `smbfs` имеет еще одно немаловажное ограничение – он не поддерживает Unicode, что может приводить к проблемам, если имена файлов содержат символы, отличающиеся от символов английского алфавита. Следует также заметить, что этот модуль ядра страдает определенными недостатками и в настоящее время больше не поддерживается. Тогда зачем его использовать? На этот вопрос легко ответить: потому что некоторые системы Linux не обладают поддержкой `cifsfs`.

После того как были сделаны необходимые предупреждения, можно попробовать смонтировать файловую систему SMB/CIFS:

```
linux:~ # mount -t smbfs //emacho/shareddocs /mnt \
-ousername=lct,password=2bblue4u,uid=jim,gid=users
linux:~ #
```

Сделать это оказалось достаточно легко! Теперь проверим работоспособность.

```
linux:~ # cd /
linux:~ # ls -ald /mnt
drwxr-xr-x 1 jim users 4096 May 20 02:50 mnt
```

Этот пример демонстрирует, что разделяемый ресурс был смонтирован локальным пользователем `jim`. Давайте скопируем какие-нибудь файлы туда и обратно:

```
linux:~ # cd /mnt
linux:~ # ls -al
total 25
drwxr-xr-x   1 lct users 4096 May 20 02:50 .
drwxr-xr-x  23 root root  560 May 18 15:21 ..
drwxr-xr-x   1 lct users 4096 Sep 30  2003 AOL Downloads
drwxr-xr-x   1 lct users 4096 May 19 12:06 CanoScanCSUv571a
dr-xr-xr-x   1 lct users 4096 Apr 16 22:42 My Music
dr-xr-xr-x   1 lct users 4096 Sep 30  2003 My Pictures
dr-xr-xr-x   1 lct users 4096 Aug  5  2004 My Videos
-rwxr-xr-x   1 lct users  129 Jul  4  2004 desktop.ini

linux:~ # cd CanoScanCSUv571a
linux:~ # ls -al
total 3451
drwxr-xr-x   1 lct users    4096 May 19 12:06 ./
drwxr-xr-x   1 lct users    4096 May 20 02:50 ../
-rwxr-xr-x   1 lct users 3398144 Mar 13  2003 CanoScanCSUv571a.exe*
-rwxr-xr-x   1 lct users   77824 Apr 26  2002 Deldrv1205.exe*
-rwxr-xr-x   1 lct users   13644 May 21  2002 N122U.cat*
-rwxr-xr-x   1 lct users   15311 May 21  2002 N122UNT.cat*
drwxr-xr-x   1 lct users    4096 May 19 12:10 N122USG/
-rwxr-xr-x   1 lct users    6151 Apr 16  2002 N122u.inf*
-rwxr-xr-x   1 lct users    8944 Jun 12  1998 USBSCAN.SYS*

linux:~ # cp USBSCAN.SYS /tmp
linux:~ # cp /var/log/messages .
linux:~ # ls -al messages
-rwxr-xr-x   1 lct users 240117 May 20 02:58 messages
```

Результаты проверки можно считать удовлетворительными, все работает. Нам удалось скопировать файл из разделяемого ресурса SMB/CIFS. Кроме того, мы скопировали файл из файловой системы Linux в разделяемый ресурс. Дополнительно существует возможность создавать, переименовывать и удалять файлы в смонтированной файловой системе SMB/CIFS. Перечень допустимых операций определяется правами доступа, которыми обладает эффективный пользователь на стороне сервера SMB/CIFS, а средства управления разрешениями файловой системы Linux контролируют доступ пользователей к точке монтирования.

Теперь, прежде чем перейти к рассмотрению использования инструментов `smbfs` из командной строки, размонтируем файловую систему:

```
linux:~ # cd /
linux:~ # df /mnt
Filesystem            1K-blocks     Used Available Use% Mounted on
//emacho/shareddocs     39061504  6782976  32278528  18% /mnt
linux:~ # umount /mnt
```

В состав дистрибутива с исходными текстами Samba входит ряд инструментальных средств, предназначенных для запуска из командной строки. Программа *smbmount* запускается командой *mount*, когда она вызывается с параметром *-t smbfs*, как это было продемонстрировано выше. Для выполнения фактической операции монтирования программа *smbmount* вызывает *smbmnt*. Процесс *smbmount* продолжает работу на протяжении всего времени, пока разделяемый ресурс остается смонтированным, и если запустить команду *ps ax*, можно будет

увидеть все имеющиеся процессы *smbmount*, каждый из которых будет соответствовать своему смонтированному ресурсу.

Программа *smbmount* считывает параметры настройки из файла *smb.conf*, хотя в нем содержится не так много информации. Фактически существует возможность использовать пустой файл с настройками и даже вообще обойтись без него! Важно обеспечить наличие файла с настройками в правильном местоположении, иначе вы будете получать сообщения об ошибках.

Подробнее о создании и проверке файла с настройками будет рассказываться далее в этой главе, а пока приведем пример файла *smb.conf* с минимально необходимым содержимым:

```
[global]
workgroup = NAME
```

Просто замените слово *NAME* именем вашей рабочей группы, которое используется в настройках Windows-систем в вашей сети.

Монтирование разделяемых ресурсов с помощью `smbmount` выполняется очень просто. Эта команда имеет следующий синтаксис:

```
smbmount UNC_resource_name mount_point options
```

где *mount_point* определяет каталог, который будет использоваться в качестве точки монтирования, как и в команде `mount`.[1] Параметр *UNC_resource_name* – это имя разделяемого ресурса в формате универсального соглашения о порядке именования ресурсов Windows (Windows Universal Naming Convention, UNC), за исключением того, что символы обратного слэша заменяются символами прямого слэша. Например, если необходимо смонтировать ресурс с именем *mydocs*, расположенный на компьютере с именем *maya*, в каталог */windocs*, следует запустить такую команду:

```
linux:~ # smbmount //maya/mydocs/ /windocs
```

Если для доступа к ресурсу потребуется ввести имя пользователя и пароль, программа *smbmount* выведет приглашение к вводу.

Теперь перейдем к более сложному примеру использования команды *smbmount*:

```
linux:~ # smbmount //maya/d /maya-d/ \
-o credentials=/etc/samba/pw,uid=jay,gid=jay,fmask=600,dmask=700
```

В этом примере для указания параметров монтирования разделяемого ресурса был использован ключ `-o`. Если двигаться по строке параметров слева направо, сначала указывается файл с регистрационными данными, который содержит имя пользователя и пароль, необходимые для получения доступа к ресурсу. Благодаря этому исключается необходимость вводить их всякий раз, когда монтируется ресурс. Формат файла с регистрационными данными очень прост:

```
username=ИМЯ_ПОЛЬЗОВАТЕЛЯ
password=ПАРОЛЬ
```

---

[1] В Linux этот каталог должен уже существовать (быть создан ранее), хотя в некоторых других UNIX-системах это необязательно: отсутствующий каталог будет создан в процессе монтирования. – *Примеч. науч. ред.*

где следует заменить *ИМЯ_ПОЛЬЗОВАТЕЛЯ* и *ПАРОЛЬ* именем пользователя и паролем, которые необходимы для аутентификации на сервере Windows рабочей группы или домена. Параметры `uid` и `gid` определяют идентификаторы владельца и группы, которые будут применяться к файлам, находящимся на разделяемом ресурсе, точно так же, как это делалось при монтировании раздела MS-DOS в предыдущем разделе. Единственное отличие состоит в том, что здесь допускается использовать как имя пользователя и имя группы, так и их числовые идентификаторы. Параметры `fmask` и `dmask` используются в качестве масок битов прав доступа, которые складываются по И с битами прав доступа, имеющимися в системе, обслуживающей разделяемый ресурс. Дополнительную информацию об этих параметрах и о том, как их использовать, вы найдете на страницах справочного руководства `smbmount`(8).

Одна из проблем, обнаруживающихся при работе с *smbmount*, состоит в том, что когда попытка смонтировать разделяемый ресурс терпит неудачу, программа ничего не сообщает о возможных причинах. В таких ситуациях гораздо удобнее пользоваться программой *smbclient*, как мы видели выше. За дополнительной информацией обращайтесь к страницам справочного руководства *smbclient*(1).

В случае, когда удаленный ресурс монтируется с помощью *smbmount* без каких-либо проблем, можно добавить соответствующую запись в файл */etc/fstab*, чтобы выполнять монтирование автоматически во время загрузки системы. Для этого достаточно использовать те же самые параметры монтирования, которые передавались команде *smbmount*, например (все параметры вводятся в одной строке):

```
//maya/d /maya-d smbfs
credentials=/etc/samba/pw,uid=jay,gid=jay,fmask=600,dmask=700 0 0
```

Итак, для краткого обзора здесь было рассмотрено достаточно много сведений. Перейдем к следующему разделу, где попробуем работать с драйвером ядра `cifsfs`, пришедшим на смену `smbfs`.

Организация доступа к Windows с помощью модуля ядра cifsfs

Драйвер файловой системы `cifsfs`, пришедший на смену драйверу `smbfs`, появился относительно недавно. В отличие от своего предшественника, `cifsfs` обладает поддержкой символов Unicode в именах файлов и каталогов. Этот новый драйвер поддерживается действующей командой разработчиков.

Если вы удостоверились в наличии поддержки модуля `cifsfs` в ядре, как было описано выше, можно попробовать смонтировать удаленный ресурс следующей командой:

```
linux:~ # mount -t cifs -ouser=lct,password=2bblue4u,uid=lct,gid=users \
//emacho/shareddocs /mnt
linux:~ # ls -ald /mnt
drwxrwxrwx  1 lct users 0 May 19 12:04 /mnt
```

Если сравнить параметры монтирования с применявшимися при использовании драйвера *smbfs*, можно заметить, что название параметра `username` изменилось на `user`. Остальные параметры остались теми же самыми.

При выводе списка содержимого каталога можно отметить одно очевидное отличие:

```
linux:~ # ls -al /mnt/CanoScanCSUv571a/
total 3684
drwxrwxrwx  1 lct users        0 May 20 02:58 .
drwxrwxrwx  1 lct users        0 May 19 12:04 ..
-rwxrwSrwt  1 lct users  3398144 Mar 13  2003 CanoScanCSUv571a.exe
-rwxrwSrwt  1 lct users    77824 Apr 26  2002 Deldrv1205.exe
-rwxrwSrwt  1 lct users    13644 May 21  2002 N122U.cat
-rwxrwSrwt  1 lct users    15311 May 21  2002 N122UNT.cat
drwxrwxrwx  1 lct users        0 May 19 12:10 N122USG
-rwxrwSrwt  1 lct users     6151 Apr 16  2002 N122u.inf
-rwxrwSrwt  1 lct users     8944 Jun 12  1998 USBSCAN.SYS
-rwxrwSrwt  1 lct users   240117 May 20 02:58 messages
```

Обратите внимание: на этот раз в поле размера для каталогов указывается число 0. Кроме этой незначительной особенности, порядок использования *cifsfs* для монтирования разделяемых ресурсов SMB/CIFS практически не отличается, за исключением случаев, когда в именах файлов встречаются многобайтовые символы (Unicode).

Команда, используемая для монтирования файловой системы CIFS/SMB (*mount -t cifs*), фактически запускает на исполнение двоичный файл *mount.cifs*. Этот файл собирается из исходных текстов пакета Samba, о чем уже говорилось выше в этой главе. В отличие от драйвера ядра `smbfs` и комплекта инструментов *smbmount*, входящих в состав дистрибутива Samba, для `cifsfs` нет никаких инструментальных средств командной строки.

Некоторые сетевые администраторы взяли себе за правило никогда не указывать пароль в параметрах утилит командной строки UNIX, поскольку это сопряжено с риском для системы безопасности. Для подобных случаев программа *mount.cifs* предусматривает два альтернативных варианта получения имени пользователя и пароля: из файла, имя которого хранится в переменной окружения `PASSWD_FILE`, и из переменных окружения `USER` и `PASSWD`. Переменная `USER` может содержать имя пользователя для нужд аутентификации на удаленном сервере. Кроме того, в эту переменную можно записать имя пользователя и пароль в формате *username%password*. Пароль пользователя может также содержаться в переменной окружения `PASSWD`. Переменная `PASSWD_FILE` может содержать путь к файлу с паролем. Программа `mount.cifs` читает из этого файла единственную строку и использует ее в качестве пароля.

Если вам когда-либо придется хранить в файле пароли в открытом виде, этому файлу необходимо будет дать чрезвычайно ограниченные права на доступ. Предпочтительнее, если возможность доступа к таким файлам будет только у процессов, для которых эти файлы предназначены.

Имя пользователя и пароль также могут храниться в файле. Имя этого файла может передаваться как часть параметра *-o* в виде *credentials=filename*. Команда *mount -t cifs* может принимать большое число дополнительных параметров, за дополнительной информацией по этому вопросу обращайтесь к страницам справочного руководства *moint.cifs*.

Использование настольных инструментов Linux совместно с libsmbclient

Офисные служащие, постоянно пользующиеся приложением Проводник в операционной системе Windows, нередко чувствуют себя потерянными, когда впервые сталкиваются с рабочим столом Linux. Это совершенно не удивительно, потому что внешний вид рабочего стола Linux несколько отличается. Инструменты называются по-другому, но это не означает отсутствие возможностей. Фактически, благодаря тому, что библиотека *libsmbclient* включается в состав всех дистрибутивов, менеджеры файлов рабочего стола в Linux (как и веб-броузеры) имеют возможность просматривать сеть Windows.

В дистрибутивах Red Hat Linux и Novell SUSE Linux поддержка возможности просмотра сети уже включена в инструменты рабочего стола. Среда окружения позволяет просматривать сеть Windows и смонтированные ресурсы NFS. Степень интеграции выше всяких похвал. Достаточно лишь щелкнуть на ярлыке просмотра сети Windows, и библиотека *libsmbclient* сделает всю черную работу. Опробуем эту возможность в окружениях рабочего стола KDE и GNOME.

В дистрибутиве Novell SUSE Linux Professional по умолчанию на рабочем столе имеется ярлык с названием Network Browsing (Обзор локальной сети). После щелчка на этом ярлыке запускается приложение Konqueror, в котором выводятся ярлыки для каждого типа сетевых ресурсов. По умолчанию в перечень ярлыков входят FTP, SLP Services (службы SLP), File Browsing SSH (просмотр файлов по протоколу SSH), SSH Nerminal (терминал SSH), VNC Connection (подключение по протоколу VNC), Windows Network (сеть Windows) и YOU Server (сервер YOU), а также ярлык Add a Network Folder (Добавить сетевую папку). После щелчка на ярлыке SMB Share (Ресурсы SMB) открывается страница, где каждая рабочая группа локальной сети отображается в виде отдельного ярлыка. В нашем примере после щелчка на ярлыке с названием рабочей группы MIDEARTH будет отображена страница с ярлыками для каждого сервера, входящего в состав этой рабочей группы. Пример окна броузера приводится на рис. 15.1.

В окружении рабочего стола GNOME имеется ярлык с названием Network Browsing (Обзор локальной сети). Двойным щелчком на этом ярлыке открывается окно обзора локальной сети, в котором присутствует ярлык Windows Network (Сеть Windows). После щелчка на этом ярлыке открывается окно, где каждая рабочая группа или домен, присутствующие в локальной сети, отображаются в виде отдельного ярлыка, как показано на рис. 15.2. После щелчка на ярлыке сервера откроется окно со списком доступных разделяемых ресурсов, после щелчка на ярлыке разделяемого ресурса – окно со списком файлов. Если доступ к какому-либо ресурсу требует аутентификации пользователя, будет выведен диалог регистрации. Пример диалога регистрации приводится на рис. 15.3.

Броузер KDE Konqueror четко и ясно отображает URL в поле адреса. По мере перемещения вглубь файловой системы Windows адрес обновляется, отражая полный эффективный URL, указывающий на текущее местоположение в сети, например *smb://alexm@MERLIN/archive/Music/Haydn/*. Синтаксис URL на страницах справочного руководства для *libsmbclient* определяется так:

```
smb://[[[domain:]user[:password@]]server[/share[/path[/file]]]][?options]
```

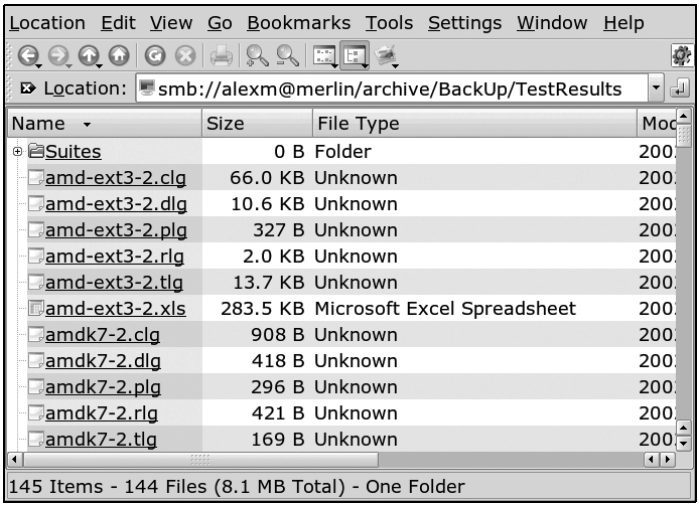


Рис. 15.1. *KDE Konqueror, использующий модуль libsmbclient*

Когда компонент *libsmbclient* вызывается приложением, он отыскивает каталог с именем *.smb* в каталоге $*HOME*, определяемый средой окружения командной оболочки пользователя. В этом каталоге отыскивается файл *smb.conf*, и если таковой будет найден, содержимое системного файла */etc/samba/smb.conf* приниматься во внимание не будет. Если данный файл отсутствует, но вместо него будет найден файл с именем *~/.smb/smb.conf.append*, то сначала будет прочитан системный файл */etc/samba/smb.conf*, а затем к его параметрам будут добавлены параметры из файла *~/.smb/smb.conf.append*.

Компонент *libsmbclient* проверяет среду окружения командной оболочки на наличие переменной окружения `USER` и будет использовать ее значение в случае отсутствия параметра `user` в строке URL.

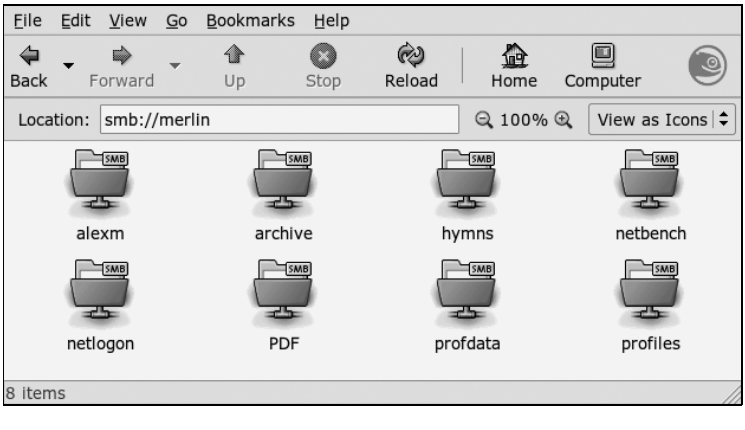


Рис. 15.2. *Менеджер файлов рабочего стола GNOME, использующий модуль libsmbclient*

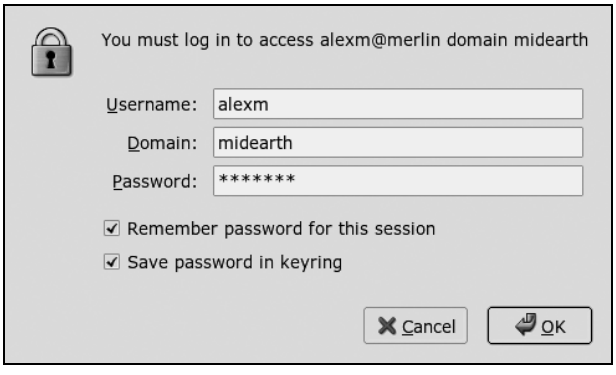


Рис. 15.3. *Диалог регистрации пользователя libsmbclient для получения доступа к сетевому ресурсу в GNOME*

Возможность библиотеки *libsmbclient* организовать аутентификацию доступа к удаленным ресурсам CIFS/SMB для каждого пользователя в отдельности – одно из существенных преимуществ. Каждое соединение (сеанс SMB) выполняется независимо, а доступ к файлам и папкам предоставляется точно так же, как если бы пользователь регистрировался на машине Windows, чтобы выполнить необходимые операции.

Печать на принтерах, подключенных к системам Windows

В прежние годы протокол Samba представлял собой единственный механизм, позволяющий выполнять печать из систем UNIX/Linux на принтеры, подключенные к компьютеру с операционной системой Windows, с использованием *smbclient* через сценарий *smbprint*. Этот сценарий по-прежнему распространяется в составе дистрибутива с исходными текстами Samba в каталоге *examples/printing* и продолжает использоваться даже при том, что на смену ему пришла утилита *smbspool*.

Когда сценарий *smbprint* только появился, доминирующими системами печати в мире UNIX/Linux были BSD *lpr/lpd* и AT&T SYSV. Позднее появился более новый инструмент с названием LPRng, который пытались продвинуть на рынок. Пакет LPRng представлял собой решение для печати, распространявшееся с открытыми исходными текстами, которым стремились заменить устаревшую технологию BSD *lpr/lpd*, считавшуюся детской забавой и требующей замены. В мире существует немало систем UNIX и Linux, где используется система печати BSD *lpr/lpd* или LPRng. В некоторых областях LPRng обладает сильными позициями. Системы, использующие LPRng, имеют тенденцию по-прежнему использовать сценарий *smbprint*, позволяющий отправлять задания печати из буфера печати UNIX/Linux на удаленные принтеры Windows.

Начиная примерно с 2000–2001 года стала обретать известность новая технология. Новый пакет получил название CUPS (Common UNIX Printing System – универсальная система печати UNIX). Рост популярности CUPS был просто фантастическим. Тем временем команда разработки CUPS постепенно расширила ее функциональные возможности и степень пригодности. Разработчики создали прикладной программный интерфейс печати и тесно сотрудничали со многими

проектами с открытыми исходными текстами, что дало возможность обеспечить высокую степень интеграции с каждым из программных проектов, где требовался доступ к системе печати. В том числе разработчики CUPS сотрудничали и с командой проекта Samba, в результате чего была реализована методология прямого сопряжения, позволившая инструментальным средствам Samba взаимодействовать с системой печати CUPS без привлечения промежуточных сценариев и утилит. Инструменты Samba могут отправлять задания печати по каналу прямо демону управления буфером печати CUPS – `cupsd`.

В дополнение к улучшенному интерфейсу между Samba и CUPS, система печати CUPS обладает намного более широкими возможностями, чем ее предшественницы, при передаче заданий печати удаленным принтерам Windows. А в рамках проекта Samba появилась новая утилита печати (*smbspool*), которая обеспечивает взаимодействие CUPS и сервера печати Windows.

Учитывая, что система печати CUPS ныне заняла доминирующее положение в Linux, лучше всего выполнять настройки печати на принтерах, подключенных к Windows, с помощью инструментальных средств либо самой системы CUPS, либо поставляемых в составе дистрибутива. С другой стороны, всегда будут иметь место ситуации, когда подобный подход неприменим. При необходимости отправлять задания печати на принтеры Windows желательно иметь некоторое представление об имеющихся инструментальных средствах. В данном случае таким инструментом является утилита *smbspool*.

Вкратце ниже приводятся возможные варианты синтаксиса команд, распознаваемых утилитой *smbspool*:

```
smb://server[:port]/printer

smb://workgroup/server[:port]/printer

smb://username:password@server[:port]/printer

smb://username:password@workgroup/server[:port]/printer
```

Любой из этих вариантов в состоянии удовлетворить все возможные потребности. В каждом из них допустимы следующие параметры:

1. Числовой идентификатор задания печати, ныне не используется утилитой *smbpool*.
2. Имя пользователя, ныне не используется утилитой *smbpool*.
3. Строка заголовка задания печати, при передаче задания передается в виде имени удаленного файла.
4. Количество печатаемых копий. Если имя файла не указывается (параметр 6), данный аргумент не используется утилитой *smbpool*.
5. Содержит дополнительные параметры печати в виде одной строки, ныне не используется утилитой *smbpool*.
6. Содержит имя печатаемого файла. Если этот параметр не указан, данные для печати принимаются с устройства стандартного ввода.

Все параметры должны следовать в указанном порядке.

Совместное использование файлов и принтеров Linux пользователями Windows

В предыдущем разделе был описан порядок применения инструментальных средств Linux для организации доступа к разделяемым ресурсам, размещенным на рабочих станциях и серверах Windows, с использованием «родных» для Windows сетевых протоколов. Те же самые инструменты могут использоваться и в обратном направлении: для организации доступа к файлам, размещенным на сервере UNIX/Linux.

В этом разделе мы рассмотрим, как использовать пакет Samba, чтобы предоставить доступ клиентам Windows к файлам, хранящимся в системе Linux.

Протокол CIFS/SMB намного сложнее любых других протоколов доступа к разделяемым файлам. Пакет Samba обеспечивает не только совместимость с клиентами Microsoft Windows на уровне протоколов, но и совместимость с ошибками, которые присутствуют в каждом клиенте. В этом разделе мы продемонстрируем простейший вариант настройки Samba, когда многие параметры имеют значения по умолчанию.

Настройка Samba включает следующие шаги:

1. Компиляция и установка программ Samba, если они еще не установлены в вашей системе.
2. Создание файла с настройками Samba *smb.conf* и проверка его на корректность.
3. Запуск двух демонов Samba – *smbd* и *nmdb*.

В случае правильной настройки сервер Samba и разделяемые каталоги появятся в списках просмотра пользователей Windows локальной сети вместе. Обычно доступ к списку осуществляется щелчком на ярлыке Сетевое окружение на рабочем столе Windows. Пользователи этих систем смогут осуществлять чтение и запись файлов из сетевых каталогов точно так же, как они это делают на своих локальных машинах или на сервере Windows, но в соответствии с вашими настройками безопасности. Сервер Samba будет виден как еще одна Windows-машина в сети и работать будет почти идентично.

Установка Samba

Корректная сборка пакета Samba из исходных текстов может оказаться делом достаточно сложным даже для искушенных разработчиков, поэтому определенно имеет смысл воспользоваться уже скомпилированными версиями пакета, если они доступны. В большинстве случаев у администратора системы есть два возможных варианта:

1. Установить Samba из проверенного и подписанного пакета RPM или .deb.
2. Установить Samba из пакета RPM или .deb, полученного со стороны.
3. Скомпилировать и установить Samba из официального архива с исходными текстами.
4. Нанять кого-нибудь, кто сможет скомпилировать и установить Samba из исходных текстов.

Большинство дистрибутивов Linux содержат Samba, позволяя установить этот пакет, просто выбрав нужный параметр при установке Linux. Если Samba не устанавливалась вместе с системой, обычно нетрудно провести установку пакета позднее. В любом случае файлы пакета Samba обычно размещаются следующим образом:

- Демоны помещаются в каталог */usr/sbin*.
- Утилиты командной строки помещаются в каталог */usr/sbin*.
- Файлы настройки помещаются в каталог */etc/samba*.
- Файлы журналов помещаются в каталог */var/log/samba*.
- Управляющие файлы времени исполнения помещаются в каталог */var/lib/samba*.

Возможны некоторые отклонения. Например, в старых версиях файлы журналов могут находиться в */var/log*, а файлы с настройками Samba – в */etc*.

Если Samba не входит в состав вашего дистрибутива, можно загрузить исходные тексты и затем скомпилировать и установить их самостоятельно. В этом случае все файлы, входящие в Samba, устанавливаются в подкаталоги ниже */usr/local/samba*.

В любом случае можно взглянуть в перечисленные каталоги, чтобы проверить, установлена ли Samba на вашей машине, и если да, то каким образом.

Если вы – не единственный системный администратор на вашей системе Linux, будьте осторожны. Другой администратор может использовать дистрибутив исходных текстов для обновления версии, установленной из двоичного пакета, и наоборот. В таких случаях файлы могут обнаружиться в двух разных местах, и вы не сразу определите, которая из установок является активной в данный момент.

При необходимости установить Samba можно воспользоваться одним из пакетов, созданных для вашего дистрибутива, либо выполнить установку из исходных текстов. Устанавливать двоичную версию обычно удобнее, но двоичные пакеты Samba поставщиков дистрибутивов Linux обычно значительно отстают от последних разработок. Даже если в вашей системе Linux уже стоит и работает Samba, можно обновить ее до последней стабильной версии из исходных текстов.

Как получить исходные тексты. Получить исходные тексты Samba можно на веб-сайте проекта по адресу *http://www.samba.org/*. Последнюю версию Samba, находящуюся в разработке, можно извлечь из репозитория *Subversion* или с помощью утилиты *rsync*.

Разработка Samba ведется открыто. Разработчики используют репозиторий *Subversion*, куда отправляют новые исходные тексты. Существует возможность получить анонимный доступ к разным веткам репозитория *Subversion* с помощью SVNweb или клиента *Subversion*.

Если используется SVNweb, за исходными текстами следует обращаться по адресу *http://svnweb.samba.org*.

Клиент системы контроля версий *Subversion* дает намного больше свободы выбора из того, что можно делать с репозиторием: он позволяет извлекать целые деревья исходных текстов и постоянно обновлять их обычными для *Subversion* командами. Этот способ доступа обычно используется разработчиками Samba.

Чтобы загрузить исходные тексты Samba из репозитория, необходимо, чтобы в системе имелся клиент *Subversion*. Ваш дистрибутив может включать в себя все необходимое, либо вы можете загрузить исходные тексты с сайта *http://subversion.tigris.org*.

Чтобы получить анонимный доступ к репозиторию *Subversion*, необходимо выполнить следующие действия:

1. Установить свежую копию *Subversion*. Все, что действительно необходимо, – это скопировать исполняемый файл клиента *Subversion*.
2. Запустить команду:

   ```
   linux:~ # svn co svn://svnanon.samba.org/samba/trunk samba.
   ```

   которая создаст каталог с именем *samba* и поместит в него последние версии исходных текстов Samba (обычно из ветки, которая готовится к следующему выпуску). К моменту написания этих строк таковой была версия 3.1.

   Другие ветки дерева исходных текстов могут быть получены добавлением *branches/ИМЯ_ВЕТКИ* к URL. Перечень имеющихся имен веток можно найти на странице *Development* веб-сайта Samba. Типичный запрос, с помощью которого можно загрузить последнюю версию 3.0 исходных текстов, выглядит следующим образом:

   ```
   lin:~ # svn co svn://svnanon.samba.org/samba/branches/SAMBA_3_0_RELEASE samba_3
   ```

3. Всякий раз, когда у вас появится желание загрузить последние обновления исходных текстов, выполните следующую команду, находясь в каталоге Samba:

   ```
   linux:~ # svn update
   ```

Сборка пакета Samba из исходных текстов. Для установки из исходного кода зайдите на веб-сайт Samba *http://www.samba.org* и щелкните на одной из ссылок, чтобы выполнить загрузку с ближайшего к вам сайта. Вы попадете на один из зеркальных сайтов для загрузки по FTP. Самая свежая стабильная версия исходных текстов находится в файле *samba-latest.tar.gz*. В этом файле вы найдете подробные инструкции по сборке и установке Samba. Если говорить кратко, вам потребуется выполнить следующие команды:

```
linux:~ # tar xzvf samba-latest.tar.gz
linux:~ # cd samba-VERSION
linux:~ # su
Password:
linux:~ # ./configure
linux:~ # make
linux:~ # make install
```

Перед выполнением сценария *configure* зарегистрируйтесь как суперпользователь. Samba несколько более требовательна в этом отношении, чем большинство других пакетов, распространяемых с открытыми исходными текстами. После выполнения перечисленных выше команд файлы Samba окажутся в следующих каталогах:

- Исполняемые файлы – в каталоге */usr/local/samba/bin*.
- Файлы с настройками – в каталоге */usr/local/samba/lib*.
- Файлы журналов – в каталоге */usr/local/samba/log*.
- Файл smbpasswd – в каталоге */usr/local/samba/private*.
- Страницы справочного руководства – в каталоге */usr/local/samba/man*.

Вам потребуется добавить каталог */usr/local/samba/bin* в переменную окружения `PATH`, чтобы запускать утилиты Samba, не указывая полного пути к ним. Кроме того, в файл */etc/man.config* нужно будет добавить следующие две строки, чтобы команда `man` смогла найти страницы справочного руководства Samba:

```
MANPATH /usr/local/samba/man
MANPATH_MAP /usr/local/samba/bin /usr/local/samba/man
```

Настройка Samba

Следующий шаг – создание файла с настройками Samba, подходящего для вашей системы. Многие программы в дистрибутиве Samba читают файл с настройками, и хотя некоторые из них могут обойтись минимумом находящейся в нем информации (и даже пустым файлом), демоны, ответственные за совместный доступ к файлам, требуют наличия полного файла с настройками.

Имя и местонахождение файла с настройками Samba зависят от того, как компилировался и устанавливался пакет. Это можно легко выяснить с помощью команды `testparm`, о которой будет рассказываться далее в этом разделе. Обычно этот файл называется *smb.conf*, и мы будем пользоваться в дальнейшем этим именем.

Формат файла *smb.conf* напоминает формат файлов *.ini*, используемых в Windows 3.x; в нем содержатся записи типа:

```
key = value
```

При работе с Samba ключи почти всегда называют *параметрами*, или *опциями*. Параметры группируются в разделы, также называемые *станзами* (*stanzas*), или строфами, которые начинаются с меток, образуемых именем раздела, заключенным в квадратные скобки. Название раздела размещается на отдельной строке, например:

```
[stanza-name]
```

Каждый каталог или принтер, предоставляемый в совместное использование, в терминологии сетей Windows называется *разделяемым ресурсом* (*share*) или *сервисом* (*service*). Каждый сервис можно задать отдельно с помощью раздела с особым именем, но мы покажем несколько способов, как упростить файл с настройками и организовать поддержку множества сервисов всего в нескольких разделах. Специальный раздел `[global]` содержит параметры, применяемые по умолчанию ко всем сервисам и к серверу в целом. Хотя Samba понимает сотни параметров, скорее всего, вы будете использовать лишь некоторые из них, так как большинство параметров имеют разумные значения по умолчанию. Если вы хотите узнать, какие параметры вообще существуют, или ищете какой-то особый параметр, можно посмотреть страницы справочного руководства для *smb.conf*(5). Но для начала создадим файл *smb.conf* со следующим содержимым:

```
[global]
    workgroup = MIDEARTH
        printing = BSD
    wins support = yes

[homes]
        browsable = no
        read only = no

[printers]
        printable = yes
        printing = BSD
        path = /var/spool/samba
[data]
        path = /export/data
        read only = no
        map archive = no
```

Несмотря на свою простоту, этих параметров настройки вполне достаточно для большинства случаев. Мы опишем все разделы в порядке их появления в файле, чтобы вы смогли понять их назначение и сделать изменения, необходимые для вашей системы. Те части, которые, скорее всего, потребуется изменить, выделены жирным шрифтом.

Раздел `[global]` содержит параметры, настраивающие Samba в целом. Например, параметр `workgroup` определяет имя рабочей группы, к которой принадлежит сервер Samba. Имя рабочей группы `MIDEARTH` нужно заменить реальным именем группы. Если на ваших машинах Windows уже задана рабочая группа, нужно воспользоваться ее именем. Если нет – задайте здесь имя новой рабочей группы и настройте все машины Windows для его использования. Не пользуйтесь устанавливаемым Windows по умолчанию именем `WORKGROUP`, чтобы избежать конфликтов с неправильно настроенными или вообще не настроенными системами.

В качестве имени компьютера для сервера (называемого также именем NetBIOS) воспользуемся предлагаемым в Samba по умолчанию именем хоста машины. Это значит, что если полное доменное имя машины – *dolphin.example.com*, то в Windows она будет видна как *dolphin*. Убедитесь, что имя хоста вашей машины установлено в соответствии с этим правилом. При желании явно определить имя сервера Samba нужно создать строку в разделе `[global]`, как показано ниже:

```
netbios name = DOLPHIN
```

Существует возможность определить имя NetBIOS, которое будет отличаться от имени хоста, поэтому вполне допустимо дать компьютеру, например, такое имя:

```
netbios name = WHITESHARK
```

В некоторых дистрибутивах Linux в качестве имени хоста по умолчанию используется имя `localhost`. Не забудьте изменить его на корректное имя, потому что любая машина, которая имеет имя NetBIOS `LOCALHOST`, полностью непригодна для работы в сети. Это имя всегда будет разрешаться клиентами Windows как IP-адрес 127.0.0.1, а это IP-адрес самого клиента!

Параметр `encrypt passwords` сообщает Samba, что клиенты будут посылать пароли в зашифрованном виде, а не открытым текстом. Это необходимо для работы Sam-

ba с Windows 98, Windows NT Service Pack 3 и последующими версиями. Если вы пользуетесь Samba версии 3.0 или выше, то эта строка не обязательна, поскольку новые версии Samba используют зашифрованные пароли по умолчанию.

Параметр `wins support` требует от Samba действовать как сервер WINS, преобразуя имена компьютеров в IP-адреса. Это не обязательно, но способствует более эффективной работе сети, о чем уже говорилось в разделе «Протоколы и другие особенности, имеющие отношение к Windows» выше в этой главе. Сервер WINS – это своего рода сервер DNS для имен NetBIOS, с той лишь разницей, что клиенты сами регистрируют себя на сервере WINS.

Остальные разделы в нашем примере файла с настройками *smb.conf* являются необязательными и описывают параметры разделяемых ресурсов Samba.

Раздел `[homes]` определяет метасервис, настраивая Samba на автоматическое выделение в совместное пользование домашних каталогов пользователей. При подключении клиента к серверу Samba производится поиск имени пользователя в файле Linux */etc/passwd* (независимо от того, существует ли определение одноименного сервиса) и определяется, есть ли у клиента учетная запись в системе. Если учетная запись есть и с ней связан домашний каталог, то этот каталог предлагается пользователю в качестве разделяемого ресурса. Имя пользователя станет именем этого ресурса, который у клиента Windows будет выглядеть как папка. Например, если у пользователя *diane* есть учетная запись на машине Samba, и он соединяется с сервером Samba, этот пользователь увидит свой домашний каталог, находящийся на машине Linux, как папку с именем *diane*.

Параметры в разделе `[homes]` определяют режим, в котором будут предоставляться домашние каталоги пользователей. Если необходимо, чтобы папка с именем `homes` отсутствовала в списке просмотра, следует задать параметр `browsable = no`.

По умолчанию Samba предоставляет разделяемые папки с правом доступа только на чтение. При установке `read only = no` папка и ее содержимое могут быть предоставлены клиенту в режиме чтения-записи. Устанавливая такие права доступа в определении совместного ресурса для файлов Linux, вы лишь накладываете дополнительные ограничения: файл, который разрешено только читать на сервере, не станет доступным для записи через сеть, если установить для него параметр `read only` равным `no`. Аналогично, если в системе Linux файл имеет разрешение на чтение-запись, то при обращении к нему сетевых клиентов Samba действует доступ Samba по умолчанию, разрешающий только чтение.

Samba не всегда выполняет простую задачу – заставить файловую систему UNIX выглядеть как файловая система Windows для клиентов Windows. Одно из различий между файловыми системами Windows и UNIX состоит в том, что в Windows есть атрибут файла archive, с помощью которого программы резервного копирования определяют, был ли некий файл модифицирован после предыдущего копирования. В случае инкрементного резервного копирования создаются копии только тех файлов, у которых установлен бит archive. В UNIX такая информация обычно следует из временной метки модификации файла, и прямого аналога атрибута archive нет. Samba моделирует атрибут архивации с помощью бита выполнения для владельца файла UNIX. Это позволяет программам резервного копирования Windows корректно работать с разделяемыми ресурсами Samba,

но побочный эффект состоит в том, что файлы данных выглядят в Linux-системе как выполняемые. Мы устанавливаем параметр `map archive` равным `no`, потому что предполагаем, что вас больше интересует организация правильной работы вашей Linux-системы, чем возможность выполнять резервное копирование с помощью приложений Windows.

В разделе `[printers]` Samba получает указание сделать принтеры, подключенные к системе Linux, доступными клиентам сети. Каждый раздел файла *smb.conf*, включая данный, в котором определен совместно используемый принтер, должен содержать строку `printable = yes`. Для доступа к принтеру нужно создать запись в системном файле Linux */etc/printcap*. Данный файл описывает все принтеры в системе и способ доступа к ним. Принтер будет виден пользователям в сети под именем, которое указано для него в файле *printcap*.

При использовании системы печати CUPS файл *printcap* создается автоматически и не должен изменяться администратором. В некоторых дистрибутивах Linux файл */etc/printcap* представляет собой символическую ссылку на файл */etc/cups/printcap*. Если предполагается выставить в общее пользование с помощью Samba лишь некоторые из имеющихся принтеров, символическую ссылку можно будет удалить, а затем создать свой собственный файл */etc/printcap*, в котором будут перечислены только те принтеры, к которым необходимо предоставить доступ для пользователей Windows. Лучший способ для этого (потому что он не зависит от реализации системы печати ОС) состоит в том, чтобы просто создать файл с именем */etc/samba/smbprintcap*, в котором будут перечислены принтеры, доступные клиентам Windows для совместного использования. После этого данный файл можно указать в файле с настройками *smb.conf* в разделе `[global]` в виде параметра `printcap =/etc/samba/smbprintcap`.

Если вы уже настроили принтер для работы, при выделении его в сетевой ресурс он может работать неправильно. Обычно при настройке принтера под Linux очередь печати связывается с драйвером принтера, который транслирует данные, получаемые от приложений, в коды, имеющие значение для конкретного используемого принтера. Однако у клиентов Windows есть свои драйверы принтеров, и они предполагают, что принтер удаленной системы принимает файлы необработанных данных, предназначенные для использования непосредственно принтером без какой-либо промежуточной обработки. Решение этой проблемы заключается в том, чтобы добавить для принтера еще одну очередь печати (или создать ее, если принтер еще не настроен), которая передает данные непосредственно на принтер. Иногда это называется «raw mode» (режим неформатированных данных).

Когда клиент Windows первый раз обращается к принтеру, ему требуется установить драйвер принтера для Windows. Эта процедура не отличается от установки принтера, подключенного непосредственно к машине клиента. Когда документ печатается на машине-клиенте Windows, он обрабатывается драйвером принтера, а затем посылается Samba. Samba просто помещает файл в очередь печати принтера, а система печати Linux завершает остальное. Исторически в большинстве дистрибутивов Linux устанавливается система печати в стиле BSD, поэтому мы задаем `printing = BSD`, чтобы известить Samba об использовании системы BSD. После этого Samba действует соответствующим образом, выдавая надлежащие команды системе печати. В последнее время некоторые дистрибутивы Linux стали использовать систему печати LPRng или CUPS. Если дистрибутив

использует LPRng, установите `printing = LPRNG`. Если используется CUPS, установите `printing = CUPS`, а также `printcap name = CUPS`.

Мы установили параметр `path` равным */var/spool/samba*, чтобы сообщить Samba, куда временно помещать двоичные файлы, получаемые от сетевого клиента, прежде чем они будут добавлены к очереди печати системы. При желании можно использовать другой каталог. Этот каталог должен быть общедоступен для записи, чтобы все клиенты имели доступ к принтеру. Здравомыслящий администратор возразит против такого решения, потому что довольно легко можно перехватить чье-нибудь задание печати и превратить его в троянского коня, с помощью которого затем можно будет скомпрометировать систему Linux. Решение проблемы состоит в том, чтобы установить «липкий» бит для этого каталога, тем самым разрешая изменять файлы только их владельцам. «Липкий» бит вместе с общедоступным правом на чтение и запись может быть установлен для каталога следующим образом:

```
linux:~ # chmod a+rwxt /var/spool/samba
```

Раздел `[data]` в нашем примере показывает, как открыть совместный доступ к каталогу. Следуя этому примеру, можно добавить сколько угодно разделяемых каталогов, указав для каждого свое имя раздела и значение `path`. В официальной документации Samba разделяемый каталог называется *разделяемой точкой* (*share-point*) в файловой системе Linux. Имя раздела используется в качестве имени разделяемого ресурса и показывается клиентам Windows как папка с таким именем. Как и в предыдущих разделах, мы воспользовались `read only = no`, чтобы разрешить доступ к ресурсу на чтение-запись, и `map archive = no`, чтобы для файлов не устанавливался бит выполнения. Параметр `path` сообщает Samba, какой каталог в системе Linux выделяется для совместного использования. Каталог может быть любым, но он должен существовать, и права доступа к нему должны соответствовать предполагаемому назначению. Для нашего разделяемого ресурса `[data]` каталог */export/data* имеет разрешения на чтение, запись и выполнение, установленные для пользователя, группы и прочих, поскольку это совместно используемый каталог общего значения, которым может пользоваться каждый.

Закончив создание файла *smb.conf*, выполните программу *testparm*, которая проверит наличие ошибок и несовместимостей в *smb.conf*. Если файл *smb.conf* не содержит ошибок, `testparm` должна возвратить удовлетворительные сообщения, например:

```
Linux:~ # testparm
Load smb config files from /usr/local/samba/lib/smb.conf
Processing section "[homes]"
Processing section "[printers]"
Processing section "[data]"
Loaded services file OK.
Press enter to see a dump of your service definitions
```

Если при создании файла *smb.conf* были допущены существенные ошибки, вы получите сообщения об ошибках вперемешку со строками, приведенными выше. В данном случае нам не требуется получить дамп определений сервисов, поэтому просто нажмите комбинацию Ctrl+C, чтобы завершить работу *testparm*.

Опытные администраторы документируют файлы с настройками Samba. Это может потребоваться позднее, когда будет необходимо вспомнить, почему те или

иные параметры были установлены именно таким образом. К сожалению, практика документирования файла *smb.conf* вступает в противоречие со способом, которым работает Samba. Файл очень часто перечитывается демоном `smbd`, таким образом, чем больший размер имеет файл из-за строк с описанием, тем большее влияние это может оказать на производительность системы. Решение этой дилеммы состоит в том, чтобы всегда использовать основной файл, в котором все параметры описаны, как это необходимо, а затем из него создавать рабочий файл с настройками с помощью команды:

```
linux:~ # testparm -s smb.conf.master > smb.conf
```

Из полученного файла *smb.conf* будут удалены все комментарии, и в нем будут находиться только те параметры, значения которых отличаются от значений по умолчанию. Файл будет иметь минимально возможный размер, реализуя все необходимые настройки. Следует предупредить, что из конечного файла будут удалены все макросы, и потому их, возможно, придется восстанавливать вручную. Например, строка `include = /etc/samba/%m.conf` превратится в строку `include=/etc/samba/.conf`.

Запуск сервера Samba

Основу Samba составляют три демона, два из которых необходимы всегда:

nmbd

Занимается регистрацией всех имен и обслуживанием запросов их разрешения. Это основной механизм, обеспечивающий возможность обзора сети. Обрабатывает все протоколы на базе UDP. В процессе запуска Samba демон *nmbd* должен запускаться в первую очередь.

smbd

Занимается обслуживанием всех соединений на базе протоколов TCP/IP к сервисам доступа к файлам и принтерам. Помимо этого, заведует процессом локальной аутентификации. Должен запускаться сразу же вслед за *nmbd*.

winbindd

Этот демон должен запускаться, когда сервер Samba должен выступать в роли члена домена Windows NT4 или Active Direcory. Это также необходимо, когда сервер Samba вступает в доверительные отношения с другим доменом. Демон *winbindd* проверяет файл *smb.conf* на наличие параметров *idmap id* и *idmap gid*, которые затем будут использоваться для отображения идентификаторов системы безопасности (SID) Windows. Указываемый в этих параметрах диапазон не должен находиться в противоречии с уже используемыми в системе идентификаторами. Если параметры *idmap id* или *gid* не определены, *winbindd* не будет выполнять отображение Windows SID, а аутентификация будет выполняться только на уровне аутентификации пользователей.

Существует возможность запускать *smbd*, *winbindd* и *nmbd* как обычные демоны или из *inetd*. Не пытайтесь использовать оба способа одновременно! Едва ли кто-то захочет иметь в системе по два экземпляра этих демонов, конкурирующих в борьбе за трафик и мешающих друг другу.

В случае непостоянного обслуживания, когда несколько пользователей соединяются с сервером лишь время от времени, есть смысл запускать эти службы из

сервера *inetd* или *xinetd*, для чего нужно поместить их описания в файл *inetd.conf*. Однако большинство администраторов запускают данные службы в виде обычных демонов из командной строки или с помощью файла */etc/rc.local*. Главное преимущество второго метода, когда *smbd* и *nmbd* запускаются как автономные демоны, состоит в том, что они откликаются на начальные запросы подключения немного быстрее.

Подробные сведения о параметрах командной строки можно найти на страницах справочного руководства. Обязательно прочитайте о том, под каким пользователем следует запускать Samba. Лучше всего запускать Samba с привилегиями пользователя *root*. В момент доступа к разделяемому ресурсу эффективный идентификатор процесса будет переключен в соответствии с эффективным идентификатором пользователя, но при этом демон должен иметь возможность решать некоторые задачи, которые в операционной системе Linux требуют наличия привилегий суперпользователя, например добавление пользователей и групп. Версия 3.0.11 Samba и более поздние версии допускают возможность эксплуатации с уровнем привилегий ниже, чем у пользователя *root*, однако демон *smbd* должен запускаться с правами пользователя *root*, чтобы быть в состоянии сделать это.

Когда пакет Samba включается поставщиком в состав операционной системы, процедура запуска Samba обычно определяется как настраиваемая часть, зависящая от типа платформы в целом. Поэтому желательно, чтобы вы отыскали в справочном руководстве администратора операционной системы информацию о порядке запуска Samba и изменили его в случае такой необходимости.

Запуск Samba из inetd.conf. Чтобы запустить Samba в качестве службы, просмотрите для начала содержимое файла */etc/services*. Есть ли там строка, описывающая порт 139/tcp? Если нет, тогда добавьте в файл строку:

```
netbios-ssn      139/tcp
```

То же необходимо сделать и для порта 137/udp, для которого необходима такая запись:

```
netbios-ns       137/udp
```

Если предполагается использовать NIS, NIS+ или LDAP для распространения таблиц отображения служб вместо файла */etc/services*, необходимо проверить их настройки. Однако шаги, которые только что были описаны, стоит пройти, потому что системы иногда возвращаются к использованию файла */etc/services*.

Далее необходимо добавить следующие строки в файл */etc/inetd.conf*:

```
netbios-ssn stream tcp nowait root /usr/local/samba/bin/smbd smbd
netbios-ns dgram udp wait root /usr/local/samba/bin/nmbd nmbd
```

Точный синтаксис файла настройки */etc/inetd.conf* может отличаться для разных дистрибутивов. Используйте в качестве образца существующие записи в файле *inetd.conf*.

В некоторых дистрибутивах в файле */etc/services* уже имеются записи, такие как `netbios_ns` (обратите внимание на символ подчеркивания). В этом случае необходимо отредактировать */etc/services* или */etc/inetd.conf*, чтобы привести имена служб в соответствие.

В некоторых дистрибутивах вместо *inetd* используется *xinetd*. За информацией о правилах настройки *xinetd* обращайтесь к страницам справочного руководства.

Во многих системах вам потребуется использовать параметр *interfaces* в файле *smb.conf*, чтобы указать IP-адреса и маски имеющихся сетевых интерфейсов. Запустите команду *ifconfig*, зарегистрировавшись как пользователь *root*, если эта информация вам неизвестна. Сервер *nmbd* пытается самостоятельно выяснить это при запуске, но на некоторых системах терпит неудачу.

В некоторых дистрибутивах допускается иметь в строках из *inetd.conf* не более пяти параметров. Обойти это ограничение можно, удалив пробелы между ключами и аргументами ключей (например, пишите *-fname* вместо *-f name*). Если никак не удается уложиться в это ограничение, напишите короткий сценарий, вызывающий требуемую команду, и вызывайте этот сценарий из *inetd*.

Активировав Samba в *inetd.conf*, необходимо перезапустить[1] демон *inetd*. Для этого просто пошлите ему сигнал HUP следующим образом:

```
linux:~ # killall -HUP inetd
```

Запуск демонов Samba, когда пакет был собран из исходных текстов по умолчанию. Если пакет Samba был установлен из дистрибутива с исходными текстами, вам потребуется сценарий, который будет запускать и останавливать работу демонов. Отыскать и скопировать такой сценарий можно из двоичного пакета, входящего в состав дистрибутива, при этом не забудьте проверить и исправить имена каталогов так, чтобы они соответствовали фактическому размещению установленных исполняемых файлов. В качестве альтернативы мы продемонстрируем, как создать и установить свой собственный сценарий.

При запуске из сценария демонам *smbd* и *nmbd* необходимо передавать параметр `-D`, чтобы они могли запуститься в режиме демонов.

Когда сценарий будет проверен и не останется сомнений по поводу его работоспособности, создайте соответствующие символические ссылки в каталогах */etc/rcN.d*, чтобы запускать Samba на том уровне запуска, на котором вы обычно загружаете систему, и останавливать на других уровнях.

Данная информация применима к системам, где пакет Samba компилировался локально с параметрами по умолчанию команды *configure*. Чтобы запустить сервер в виде демона, можно использовать следующий сценарий, которому можно было бы дать говорящее название *startsmb*:

```
#!/bin/sh
/usr/local/samba/bin/smbd -D
/usr/local/samba/bin/winbindd
/usr/local/samba/bin/nmbd -D
```

Не забудьте сделать сценарий исполняемым:

```
linux:~ # chmod +x startsmb
```

---

[1] Это не перезапуск `inetd`; по соглашениям `inetd` получение сигнала HUP обязывает его перечитать заново конфигурационный файл *inetd.conf*. – *Примеч. науч. ред.*

Такой сценарий можно запускать вручную или вызывать его из системного сценария *rc*.

Если в файле *smb.conf* отсутствуют ошибки, возможность отказа демонов можно считать практически невероятной. Тем не менее будет нелишним проверить наличие демонов в списке активных процессов командой `ps ax`. В противном случае сообщения об ошибках можно найти в файлах журналов Samba *log.smbd* и *log.nmbd*.

Чтобы остановить Samba, достаточно послать сигнал процессам *nmbd* и *smbd*. В Debian для этих целей можно использовать команду `killall`, с помощью которой послать сигнал SIGTERM:

```
# killall -TERM smbd nmbd
```

Управление исполнением Samba в Debian Linux. Для запуска (*start*), остановки (*stop*) и перезапуска (*restart*) Samba можно использовать сценарий *samba*:

```
linux:~ # /etc/init.d/samba start
```

Управление исполнением Samba в SUSE и Red Hat Linux. В SUSE Linux реализован индивидуальный подход к управлению исполнением каждого демона Samba. Сценарий управления исполнением Samba, который можно вызывать из командной строки, приводится в примере 15.1. Этот сценарий можно поместить в каталог */sbin*, присвоив ему имя *samba*. Такого рода управляющие сценарии должны принадлежать пользователю *root* и группе *root*, а биты разрешений должны быть установлены таким образом, чтобы исполняемым этот сценарий был только для владельца.

Пример 15.1. Сценарий управления исполнением Samba для SUSE Linux

```
#!/bin/bash
#
# Сценарий запуска/остановки samba
# Разместите этот сценарий в каталоге /sbin с именем файла 'samba'

RCD=/etc/rc.d

if [ z$1 == 'z' ]; then
    echo $0 - Отсутствует обязательный аргумент, должно быть start или stop.
    exit
fi
if [ $1 == 'start' ]; then
    ${RCD}/nmb start
    ${RCD}/smb start
    ${RCD}/winbind start
fi
if [ $1 == 'stop' ]; then
    ${RCD}/smb stop
    ${RCD}/winbind stop
    ${RCD}/nmb stop
fi
if [ $1 == 'restart' ]; then
    ${RCD}/smb stop
    ${RCD}/winbind stop
    ${RCD}/nmb stop
    sleep 5
```

```
    ${RCD}/nmb start
    ${RCD}/smb start
    ${RCD}/winbind start
fi
exit 0
```

Типичный сценарий запуска для Red Hat Linux приводится в примере 15.2. Этому файлу можно присвоить имя *smb* и поместить его в каталог */etc/rc.d*. Подобный сценарий необходим для управления работой *winbind*. Если вам нужна дополнительная информация о сценариях запуска, загляните в каталог *packaging* дерева каталогов с исходными текстами Samba. Здесь вы найдете сценарии, управляющие запуском для каждой из платформ.

Пример 15.2. Сценарий управления исполнением Samba для Red Hat Linux

```
#!/bin/sh
#
# chkconfig: 345 81 35
# description: Запускает и останавливает демоны Samba - smbd и nmbd \
#              используется для запуска сетевых служб.

# Библиотека функций.
. /etc/rc.d/init.d/functions
# Параметры настройки сети.
. /etc/sysconfig/network
# Проверить наличие подключения к сети.
[ ${NETWORKING} = "no" ] && exit 0
CONFIG=/etc/samba/smb.conf
# Проверка наличия smb.conf
[ -f $CONFIG ] || exit 0

# Проверить, как вызывался сценарий.
case "$1" in
  start)
       echo -n "Запускаются службы SMB: "
       daemon smbd -D; daemon nmbd -D; echo;
       touch /var/lock/subsys/smb
       ;;
  stop)
       echo -n "Останавливаются службы SMB: "
       smbdpids=`ps guax | grep smbd | grep -v grep | awk '{print $2}'`
       for pid in $smbdpids; do
         kill -TERM $pid
       done
       killproc nmbd -TERM; rm -f /var/lock/subsys/smb
       echo ""
       ;;
  status)
       status smbd; status nmbd;
       ;;
  restart)
       echo -n "Перезапускаются службы SMB: "
       $0 stop; $0 start;
       echo "done."
       ;;
```

```
  *)
      echo "Порядок использования: smb {start|stop|restart|status}"
      exit 1
esac
```

Проверка – запущены ли службы Samba. Теперь, после установки, настройки и запуска Samba, попробуйте выполнить команду `smbclient`, чтобы получить список разделяемых ресурсов (данные результаты были получены в офисной сети в день, когда на работе были всего два сотрудника):

```
linux:~ # smbclient -L localhost -U%
added interface ip=172.16.1.3 bcast=172.16.1.255 nmask=255.255.255.0
Domain=[MIDEARTH] OS=[Unix] Server=[Samba 3.0.20]
        Sharename       Type      Comment
        ---------       ----      -------
        archive         Disk      Full Archive Files
        print$          Disk      Printer Drivers
        netlogon        Disk      Network Logon Service
        profiles        Disk      Profile Share
        IPC$            IPC       IPC Service (Main Server)
        ADMIN$          IPC       IPC Service (Main Server)
        kyocera         Printer   FS-C5016N
Domain=[MIDEARTH] OS=[Unix] Server=[Samba 3.0.20]

        Server               Comment
        ---------            -------
        AURORA               Moberg's Magic Machine
        MERLIN               Main Server
        TINKERBELL           Mel's Laptop

        Workgroup            Master
        ---------            -------
        MIDEARTH             MERLIN
```

Полученные результаты показывают, что сервер Samba допускает возможность анонимного подключения. Анонимное подключение – это подключение, при котором не используются ни имя пользователя, ни пароль. В этом случае подключающийся пользователь получает права доступа *гостевой учетной записи* (*guest account*), которая обычно соответствует учетной записи пользователя с именем `nobody` в файле */etc/passwd*.

Если на этом шаге попытка подключения терпит неудачу, причина, скорее всего, в том, что трафик Samba блокируется системой сетевой защиты (firewall), или что гостевая учетная запись не была найдена в файле */etc/passwd*.

Добавление пользователей

Сетевые клиенты должны быть аутентифицированы Samba, чтобы получить доступ к совместным ресурсам. В конфигурации данного примера Samba использует безопасность на уровне пользователя; от клиентов требуется передать имя пользователя и пароль, которые должны соответствовать имеющимся на хосте Linux. Первый шаг в добавлении нового пользователя Samba – обеспечить наличие у этого пользователя учетной записи Linux, а если в файле *smb.conf* присутствует раздел `[homes]`, то существование у этой учетной записи домашнего каталога.

Наиболее часто добавление нового пользователя в систему Linux выполняется с помощью useradd.

```
linux:~ # useradd -m username
```

Кроме того, в Samba есть свой файл паролей, служащий для проверки зашифрованных паролей, получаемых от клиентов. Для каждого пользователя Samba нужно выполнить команду *smbpasswd*, чтобы добавить учетную запись Samba для этого пользователя:

```
linux:~ # smbpasswd -a username
New SMB password:
Retype new SMB password:
```

Следите, чтобы имя пользователя и пароль, передаваемые *smbpasswd*, были такими же, как в учетной записи пользователя Linux. Предлагаем начать с добавления собственной учетной записи, которой можно будет потом воспользоваться для тестирования правильности установки.

Офисное решение организации совместного доступа к файлам и принтерам с помощью Samba

Теперь, когда вы узнали, что такое Samba, как создать типичный файловый сервер, а также как запускать его и как останавливать, настало время перейти к настройке сервера в более сложной конфигурации. Пример, рассмотрением которого мы займемся здесь, является типичным для офисов с количеством пользователей от 5 до 50. К серверу можно обращаться из старых систем Windows, использующих устаревшие сетевые технологии на базе рабочих групп, но сервер может также служить контроллером домена Samba, который обеспечивает полностью безопасный, аутентифицированный доступ к сетевым службам. Сложность состоит не в архитектуре сервера, а в том, как он используется, и как этого достичь.

Первоочередная задача состоит в том, чтобы настроить централизованный сервер, который будет предоставлять возможность совместного доступа к файлам и принтерам.

Мы рассмотрим установку двух типов принтеров: принтер с сетевым интерфейсом и принтер с интерфейсом USB. Установка сетевого принтера выполняется достаточно просто. Типичный пример такого принтера – HP LaserJet с сетевой платой JetDirect. Этот тип принтера может быть установлен с помощью инструментов командной строки следующим образом:

```
linux:~ # lpadmin -p hplj -v socket://192.168.1.25:9100 -E
```

В этом примере сетевая плата HP JetDirect была запрограммирована на работу с IP-адресом 192.168.1.25. Система печати CUPS будет взаимодействовать с принтером непосредственно через порт TCP 9100, а параметр -E означает, что деятельность очереди печати с именем hplj активируется немедленно.

Данная команда не устанавливает драйвер принтера, поскольку таковой не был определен. Чтобы установить драйвер, нужно добавить параметр *-m модель*. В этом случае необходимо убедиться, что модель принтера указана правильно. В данном случае можно было бы добавить в конец команды параметр *-m LaserJet-laserjet*.

Принтер Canon BJC-85, подключенный к интерфейсу USB, устанавливается как очередь печати без промежуточного форматирования данных (raw printer, то есть без фильтрации печати) следующей командой:

```
linux:~ # lpadmin -p bj85 -v usb://Canon/BJC-85 -m BJC-85-bjc600 -E
```

Современные дистрибутивы Linux автоматически обнаруживают присутствие принтера, подключенного к порту USB или к параллельному порту, и запрашивают подтверждение на выполнение автоматической настройки. В большинстве случаев драйвер принтера выбирается автоматически без необходимости делать это вручную.

В Linux есть два основных способа сделать системный принтер доступным для совместного использования клиентами Windows. Первый из них известен как *режим неформатированных данных* (*raw mode*), а второй – как *интеллектуальный режим* (*smart mode*).

В режиме неформатированных данных система печати Linux просто передает задание печати на принтер без какой-либо промежуточной обработки данных. Это наиболее общий способ работы систем печати Беркли (*lpr/lpd*) и LPRng при минимальных настройках. Система печати CUPS также поддерживает возможность работы в режиме неформатированных данных. Параметр настройки Samba `cups options = raw` разрешает системе CUPS работать в этом режиме.

Если данный параметр отсутствует в файле настроек *smb.conf*, и система CUPS получит задание печати, содержащее последовательность символов, которая не известна фильтрам CUPS, задание может быть удалено и не попадет на принтер. Один из способов добиться выполнения заданий печати через CUPS (когда оказывается невозможным определить параметр `cups options = raw`) состоит в том, чтобы отредактировать файлы */etc/cups/mime.types* и */etc/cups/mime.convs*, где нужно раскомментировать строки, определяющие тип MIME `application/octet-stream`. Это позволит системе CUPS отправлять задания печати с неизвестными последовательностями символов непосредственно на принтер.

Печать в режиме неформатированных данных требует установки корректного драйвера принтера на все компьютеры с Windows. Клиент Windows должен самостоятельно выполнить всю необходимую обработку заданий печати и подготовить их к непосредственной передаче на принтер.

Для использования интеллектуального режима необходимо установить на сервере CUPS локальную систему фильтрации. Сервер CUPS будет пытаться интерпретировать тип файла, отправляемого на принтер, и затем фильтровать данные, автоматически выбирая преобразование, соответствующее принтеру.

Когда система CUPS используется в интеллектуальном режиме, допускается использовать драйвер CUPS PostScript (доступный на веб-сайте CUPS) на всех клиентах Windows, даже если принтер не совместим с форматом PostScript. Система CUPS сама будет преобразовывать задания печати в требуемый формат. Однако такой подход требует установки драйвера принтера, способного воспроизводить выходной формат, известный системе фильтрации CUPS.

В примере 15.3 приводится пример файла настроек *smb.conf*. В этом примере реализован режим печати неформатированных данных. Если драйвер принтера в системе CUPS выбран правильно, на стороне клиента Windows может использо-

ваться драйвер практически любого из принтеров, поддерживаемых Windows. Например, можно даже использовать драйвер Color LaserJet для печати на принтере Epson Bubblejet.

Пример 15.3. Пример файла настроек smb.conf для офисной сети

```
# Глобальные параметры
[global]
    workgroup = GOODOIL
    netbios name = LOUDBELL
    passwd chat = *Новый*Пароль* %n\n *Повторный*ввод*пароля* %n\n*Пароль*изменен*
    username map = /etc/samba/smbusers
    syslog = 0
    name resolve order = wins bcast hosts
    printcap name = CUPS
    cups options = raw
    show add printer wizard = No
    add machine script = /usr/sbin/useradd -s /bin/false -d /dev/null `%u`
    logon script = scripts\logon.bat
    logon path =
    logon home = \\%L\%U
    logon drive = H:
    domain logons = Yes
    preferred master = Yes
    wins support = Yes
[homes]
    comment = Домашние каталоги
    valid users = %S
    read only = No
    browseable = No
[printers]
    comment = Очередь печати SMB
    path = /var/spool/samba
    guest ok = Yes
    printable = Yes
    use client driver = Yes
    default devmode = Yes
    browseable = No
[netlogon]
    comment = Служба сетевой регистрации
    path = /var/lib/samba/netlogon
    guest ok = Yes
[officedata]
    comment = Разделяемый каталог офиса
    path = /data/office
    read only = No
```

За информацией о настройке принтеров в Linux обращайтесь к главе 14. Инструментальные средства Samba в состоянии напрямую взаимодействовать с системой печати CUPS посредством библиотеки *libcups.so*. Чтобы настроить Samba на взаимодействие с системой печати LPRng, просто замените `printcap name = CUPS` на `printcap name = LPRNG`. Все принтеры автоматически становятся доступны для использования пакетом Samba.

Скопируйте файл *smb.conf* в нужный каталог. Запустите Samba, следуя рекомендациям, данным в разделе «Запуск сервера Samba» выше в этой главе.

Создайте в файловой системе Linux каталог */data/office* и установите биты прав доступа к каталогу таким образом, чтобы пользователи Linux и Windows (Samba), обладая соответствующими привилегиями, могли иметь к нему доступ. Например, если все пользователи должны иметь доступ к каталогу на чтение, а пользователь `jamesb` сверх того должен иметь возможность записи в каталог, выполните следующие команды:

```
linux:~ # chown -R jamesb:users /data/office
linux:~ # chmod -R u=rwx,g=rx,o-rwx /data/office
```

После запуска Samba добавьте учетную запись пользователя, как описано в разделе «Добавление пользователей». Когда учетная запись будет создана, попробуйте воспользоваться командой *smbclient*, описанной ранее в разделе «Организация доступа к Windows с помощью FTP-подобного клиента smbclient»:

```
linux:~ # smbclient //localhost/officedata -U'username'
password: XXXXXXXXXX
```

Здесь *username* – это имя пользователя, а *XXXXXXXXXX* – пароль, введенный при добавлении нового пользователя с помощью команды *smbpasswd*.

После появления приглашения к вводу `smb:>` можно будет вводить любые команды *smbclient*. Попробуйте ввести команду *ls*, чтобы просмотреть содержимое каталога. Затем попробуйте ввести команду *help*, которая выведет список всех доступных команд. Программа *smbclient* по своему поведению очень близко напоминает программу *ftp*, таким образом, если вам приходилось работать с *ftp*, вы должны чувствовать себя как дома. Теперь выйдите из программы *smbclient* (используя для этого команду *quit* или *exit*) и попробуйте изменить некоторые параметры команды. Для начала вместо *localhost* укажите настоящее имя хоста сервера *loudbell*, чтобы проверить работоспособность службы имен. Затем попробуйте получить доступ к своему домашнему каталогу с использованием своего имени пользователя вместо *officedata*.

А теперь нечто более интересное: перейдите на Windows-машину и зарегистрируйтесь с именем пользователя и паролем вашей учетной записи на Samba. (В Windows NT/2000/XP нужно добавить учетную запись для нового пользователя с именем и паролем из учетной записи Samba.) Щелкните дважды на ярлыке Сетевое окружение на рабочем столе. Найдите в сети свою рабочую группу и дважды щелкните на ее ярлыке. Вы должны увидеть в открывшемся окне ярлык для своего сервера Samba. Дважды щелкнув на нем, вы откроете окно, в котором будут показаны ваш домашний каталог, принтер и разделяемый ресурс *officedata*. Теперь можно перетаскивать мышью файлы между вашим исходным каталогом и разделяемыми ресурсами, а после установки драйвера для совместно используемого принтера отправлять задания печати Windows на ваш принтер Linux!

Мы коснулись лишь малой части того, что может делать Samba, но и по ней можно получить представление, почему Samba способствовала известности Linux, хотя этот пакет создавался не только для Linux.

Автоматическая загрузка драйвера принтера

Сетевые администраторы Windows понимают преимущества простоты установки принтеров на рабочих станциях Windows. Рассмотрим, к примеру, как действует администратор, когда драйверы принтеров не были размещены на сервере печати. Сетевой администратор приходит на рабочее место пользователя, которому нужно настроить доступ к новому принтеру, начинает устанавливать принтер и обнаруживает, что забыл диск с драйверами на своем столе. Теперь он должен возвращаться к своему рабочему месту, а в некоторых крупных фирмах это может оказаться довольно приличное расстояние. Придя к пользователю еще раз, он обнаруживает, что драйвер не работает и необходима более новая версия драйвера. Гораздо удобнее иметь все драйверы принтеров уже установленными на сервере печати!

Проблема здесь состоит в том, что задача, которая для одних – лишь часть святой чаши Грааля сетевого администрирования, для других может оказаться непосильной. Пошаговая инструкция, которая приводится ниже, поможет вам настроить возможность автоматической установки драйверов принтеров прямо с сервера печати. Читайте рекомендации очень внимательно и неуклонно следуйте им, потому что одна оплошность может стать причиной неудачи.

Сначала нужно изменить файл *smb.conf*, как показано в примере 15.4. Изменения по сравнению с предыдущим листингом выделены жирным шрифтом.

Пример 15.4. Пример файла настроек smb.conf для офисной сети

```
# Глобальные параметры
[global]
    workgroup = TOPCAT
    netbios name = LOUDBELL
passwd chat = *Новый*Пароль* %n\n *Повторный*ввод*пароля* %n\n*Пароль*изменен*
    username map = /etc/samba/smbusers
    syslog = 0
    name resolve order = wins bcast hosts
    printcap name = CUPS
    cups options = raw
    show add printer wizard = Yes
    add machine script = /usr/sbin/useradd -s /bin/false -d /dev/null `%u`
    logon script = scripts\logon.bat
    logon path =
    logon home = \\%L\%U
    logon drive = H:
    domain logons = Yes
    printer admin = jbloggs
    preferred master = Yes
    wins support = Yes
[homes]
    comment = Домашние каталоги
    valid users = %S
    read only = No
    browseable = No
[printers]
    comment = Очередь печати SMB
```

```
    path = /var/spool/samba
    guest ok = Yes
    use client driver = No
    printable = Yes
    default devmode = Yes
    browseable = No
[print$]
    comment = Драйвер принтера
    path = /var/lib/samba/drivers
[netlogon]
    comment = Служба сетевой регистрации
    path = /var/lib/samba/netlogon
    guest ok = Yes
[officedata]
    comment = Разделяемый каталог офиса
    path = /data/office
    read only = No
```

Когда файл *smb.conf* будет отредактирован, проверьте, запущен ли сервер Samba.

В нашем примере файла *smb.conf* мы определили, что правом управления принтерами наделяется пользователь Windows *jbloggs*.

На следующем шаге необходимо создать каталог */var/lib/samba/drivers*, а также все подкаталоги, вложенные в него. В этом каталоге будут размещаться файлы драйвера принтера для Windows. Для этого достаточно выполнить следующие команды:

```
linux:~ # mkdir -p /var/lib/samba/drivers
linux:~ # cd /var/lib/samba
linux:~ # mkdir -p drivers/{W32ALPHA,W32MIPS,W32PPC}
linux:~ # mkdir -p drivers/{W32X86/{2,3},WIN40,COLOR,IA64,x64}
linux:~ # chown -R jbloggs:root drivers
linux:~ # chmod -R u+rwx,g+rwx,o+rx-w drivers
```

Установите в системе Linux принтеры, которые необходимо сделать доступными для клиентов Windows. Совершенно не важно, какая система печати будет использоваться. Вполне подойдет система LPRng, хотя система CUPS обладает большим количеством разного рода «бантиков и рюшек». В любом случае желательно будет использовать печать в режиме неформатированных данных даже там, где это ранее не планировалось. Использование режима неформатированных данных поможет избежать неприятностей, когда какой-нибудь драйвер принтера, установленный на стороне клиента Windows, будет создавать такие задания печати, которые не смогут быть обработаны фильтрами CUPS.

Теперь все готово к установке драйвера принтера на сервер Samba. Следующая пошаговая процедура поможет обеспечить доступность драйверов принтеров.

1. Зарегистрируйтесь на рабочей станции Windows XP в качестве пользователя `jbloggs`.
2. Щелкните на ярлыке Сетевое окружение правой кнопкой мыши и выберите пункт контекстного меню Проводник.
3. Откройте сначала ветку Вся сеть, а затем Microsoft Windows Network. Выберите рабочую группу или домен, где находится сервер Samba. Щелкните на элементе

списка, соответствующем серверу Samba (в данном случае это компьютер с именем TOPCAT).

4. Щелкните на ярлыке Принтеры и факсы. В правой части окна программы Проводник вы должны увидеть принтеры, доступные через систему печати Linux.
5. Щелкните правой кнопкой мыши на ярлыке принтера, драйвер которого требуется установить. На экране появится сообщение: «Драйвер принтера не установлен на этом компьютере. Некоторые свойства принтера будут недоступны до тех пор, пока не будет установлен драйвер принтера. Установить этот драйвер?». На выбор будет предложено два варианта ответа: Да (в некоторых системах Продолжить) и Нет. Щелкните на кнопке Нет. Не нужно щелкать на кнопке Да (или Продолжить).
6. Выберите вкладку Дополнительно в появившемся диалоге свойств принтера.
7. Щелкните на кнопке Сменить.... Перед вами откроется диалог выбора драйвера принтера. Выберите производителя и модель принтера. Если установку драйвера необходимо выполнить с компакт-диска или с сетевого ресурса, щелкните на кнопке Установить с диска....
8. Следуйте подсказкам в следующих диалогах. Внимательно следите за процессом установки драйверов – они должны отправляться на сервер Samba (в нашем случае на сетевой ресурс *\\TOPCAT\print$\W32X86*). По окончании диалог можно закрыть и поздравить себя.

Как побочный эффект процесса установки драйвера на сервер, драйвер одновременно будет установлен на рабочей станции, которая использовалась для этого. Когда в следующий раз вам придется устанавливать драйвер на рабочей станции Windows XP, достаточно будет просто щелкнуть на ярлыке принтера в окне Сетевое окружение, и драйвер будет установлен без запроса на разрешение выполнения установки драйвера.

Использование smbsh для выполнения непосредственных операций над файлами в удаленной системе

Утилита *smbsh* позволяет манипулировать файлами на удаленной системе, используя стандартные команды UNIX или Linux. Для работы с этой командной оболочкой необходимо запустить команду smbsh из строки приглашения к вводу и ввести имя пользователя и пароль, идентифицирующие вас на машине, работающей под управлением операционной системы Windows NT. Запуск выглядит следующим образом:

```
system$ smbsh
Username: user
Password: XXXXXXX
```

После этого можно будет вводить команды на удаленной системе, как если бы это была локальная система. Например, команда *ls /smb* выведет список рабочих групп, а команда *ls /smb/MYGROUP* – список всех машин в рабочей группе MYGROUP. Команда *ls /smb/MYGROUP/machine-name* выведет список имен разделяемых ресурсов, доступных на заданной машине. Для перехода из каталога в каталог можно использовать команду *cd*, команду *vi* – для редактирования файлов и *rcp* – для копирования файлов.

Работоспособность *smbsh* зависит от механизма динамического связывания библиотек, который известен как механизм *предварительной загрузки*. Данная утилита использует предварительно загруженную библиотеку с именем *smbwrapper.so*. Эта библиотека перехватывает обращения к функциям файловой системы и передает их библиотеке CIFS/SMB, если файлы, над которыми выполняются операции, находятся в пределах каталога */smb*. (Если файл находится за пределами каталога */smb*, обращения к функциям файловой системы будут передаваться стандартной системной библиотеке, как если бы данной библиотеки не существовало вовсе.) Таким образом, любая команда, выполняемая из командной оболочки *smbsh* и обращающаяся к каталогу */smb*, будет использовать протокол SMB.

В дистрибутиве с исходными текстами Samba версии 3 имеются две отдельные реализации *smbsh*. Одна из них компилируется из исходных текстов в каталоге *source*. Другая находится в каталоге *examples*. Версия в каталоге *source* – это оригинальная самостоятельная реализация, которая больше не работает в операционной системе Linux, но продолжает использоваться на таких традиционных платформах, как Sun Solaris, HP-UX и AIX. Неработоспособность этой версии в системе Linux обусловлена решением, принятым командой сопровождения *glibc*, изменить поведение библиотеки, начиная с версии 2.1.

Реализация утилиты *smbsh*, расположенная в каталоге *examples*, вполне работоспособна, хотя имеет ошибку. Данная версия использует библиотеку *libsmbclient*. Ошибка, содержащаяся в этой реализации *smbsh*, проявляется в том, что получить список файлов с помощью команды *l*s можно только извне виртуального каталога */smb*, который создается утилитой. Пока нет точных сведений, когда эта ошибка будет исправлена командой разработки Samba.

Несмотря на мелкие недостатки, имеющиеся в *smbsh*, она продолжает использоваться множеством приложений, которые изначально не поддерживают CIFS/SMB, но нуждаются в такой поддержке. Этот инструмент, возможно, единственный способ организации поддержки CIFS/SMB для приложений, которые не могут быть обновлены для непосредственного использования библиотеки *libsmbclient*.

Настройка NFS и NIS

Установив TCP/IP, можно настроить машину на использование NFS (Network File System – сетевая файловая система) или NIS (Network Information Service – сетевая информационная служба). NFS позволяет вашей системе напрямую совместно использовать файлы с некоторой сетью машин. Доступ к файлам в NFS прозрачен: вы можете использовать их так, как если бы они находились на локальном диске. В терминах сетевого администрирования это выглядит так: одна система монтирует удаленную файловую систему в локальный каталог так же, как может быть монтирована локальная файловая система. NFS также позволяет экспортировать файловые системы, давая возможность другим машинам в сети напрямую монтировать ваши диски.

NIS – это система, позволяющая вашему хосту автоматически получать от сетевых серверов информацию об учетных записях пользователей, групп, точках монтирования файловой системы и других системных базах данных. Допустим, к примеру, что у вас есть много машин, которые должны иметь одинаковые учетные записи пользователей и групп (информация, обычно находящаяся в файлах

/etc/passwd и */etc/group*). Пользователи должны иметь возможность заходить на любую из этих машин и напрямую получать доступ к их файлам (скажем, смонтировав файловую систему своего исходного каталога с центрального сервера при помощи NFS). Очевидно, управлять учетными записями на множестве машин не так просто. Например, для того чтобы добавить нового пользователя, необходимо зайти по очереди на все машины и создать на каждой из них учетную запись. Однако при использовании NIS система автоматически запрашивает такую информацию по сети у централизованно управляемой базы данных, а не только ищет ее в локальных файлах */etc/passwd*. NIS+ – это расширенная версия службы NIS, которую уже начинают использовать.

NFS имеет две стороны. Одна из них дает возможность экспортировать части файловой системы сервера или рабочей станции, чтобы другие пользователи могли обращаться к файлам и каталогам, а вторая позволяет монтировать удаленные ресурсы на локальной рабочей станции или сервере, что дает возможность обращаться к ним как к локальным файловым системам. Ресурсы NFS экспортируются сервером NFS. Смонтированные локально ресурсы NFS будут доступны на клиенте NFS.

Следует предупредить, что NFS не обеспечивает абсолютно никакого шифрования. Если файловые системы монтируются через Интернет, существует вероятность, что передаваемые файлы могут быть изменены злоумышленником в любой момент времени (существует даже такая шуточная расшифровка аббревиатуры NFS: «No File Security» – полное отсутствие защиты файлов). С другой стороны, скорость работы с разделами NFS, смонтированными за пределами локальной сети, скорее всего, окажется слишком низкой, чтобы иметь какую-то практическую ценность, если только вы не являетесь счастливым обладателем очень широкого канала.

Если вашей системе Linux предстоит взаимодействовать с другими системами в локальной сети, вполне возможно, что NFS и NIS уже широко используются в локальной сети. В этом разделе мы покажем, как настроить систему для работы в качестве клиента NFS и NIS, то есть монтировать удаленные файловые системы и участвовать в существующем домене NIS. Можно превратить вашу систему и в сервер NFS, и в сервер NIS, но в процедуре настройки UNIX- или Linux-систем в качестве NFS/NIS-сервера существует множество тонких моментов. Вместо того чтобы давать здесь неполные, и потому опасные, советы по настройке сервера, мы отсылаем вас к книге «Managing NFS and NIS» Хала Штерна (Hal Stern), изданной O'Reilly. Если вы уже знакомы с настройкой NFS/NIS на других системах UNIX, то в Linux вы найдете не много отличий. Все особенности вы найдете на страницах справочного руководства и в разнообразных документах Linux HOWTO.

Настройка системы для работы в качестве клиента NFS

Несколько слов предостережения по поводу NFS. Прежде всего, NFS не любит, когда серверы с удаленными файловыми системами выключаются или нарушается сетевое соединение. Когда по какой-то причине сервер NFS недоступен, ваша система периодически выводит предупреждающие сообщения на консоль (или в системные журналы). Если такое происходит, можно попробовать размонтировать удаленную файловую систему недоступного сервера командой *umount* (глава 10).

Другое, на что следует обратить внимание при монтировании файловых систем NFS, – это идентификаторы владельца (*uid*) и группы (*gid*) файлов на удаленной файловой системе. Для доступа через NFS к собственным файлам числовые идентификаторы пользователя и группы вашей учетной записи должны совпадать с аналогичными идентификаторами на сервере NFS. Простой способ проверить это – выполнить команду *ls -l*. Если *uid* или *gid* не совпадают с каким-либо локальным пользователем, *ls* выведет *uid*/*gid* файлов в виде чисел, а в случае совпадения будет показано имя пользователя или группы.

Если числовые идентификаторы не совпадают, возникшую проблему можно будет решить несколькими способами. Можно просто изменить *uid* вашей учетной записи (и *gid* основной группы) так, чтобы они совпадали с записями на сервере NFS (отредактировав ваш локальный файл */etc/passwd*). Тогда после внесения изменений потребуется выполнить команды *chown* и *chgrp* для всех локальных файлов. Другое решение – создать отдельную учетную запись с подходящим *uid*/*gid*. Однако лучше всего, вероятно, воспользоваться NIS для управления базами данных пользователей и групп. При таком подходе вам не надо создавать локальные учетные записи, вместо этого они предоставляются сервером NIS. Далее мы расскажем об этом подробнее.

Другим тонким местом NFS является ограничение разрешений для *root* на файловых системах, монтируемых через NFS. Если только NFS-сервер явно не предоставил доступа в качестве *root* к файловым системам NFS, у вас не будет полного доступа к файлам при регистрации в качестве суперпользователя на вашей локальной системе. Это сделано в целях безопасности: предоставление суперпользователю неограниченного доступа к файлам на удаленных файловых системах NFS открывает возможности для злоупотреблений, особенно когда клиент и сервер NFS принадлежат или управляются разными людьми.

Монтирование экспортируемых ресурсов NFS на стороне клиента может выполняться разными способами:

- Автоматически, во время монтирования файловых систем, описываемых файлом */etc/fstab* на этапе загрузки.
- Вручную из командной строки.
- Автоматически с помощью демона автомонтирования.

Здесь мы не будем рассматривать вопрос автомонтирования, поскольку это выходит за рамки данной главы. Однако необходимую информацию вы найдете в главе 10. В следующих разделах будет дан краткий обзор остальных двух методов.

Использование записи в файле /etc/fstab на стороне клиента NFS

Настроить систему для монтирования удаленных файловых систем с помощью NFS легко. Предположим, что протоколы TCP/IP настроены и распознавание имен хостов работает правильно. Тогда можно просто добавить в файл */etc/fstab* строку следующего вида:

```
# device       directory           type   options    dump fsckorder
allison:/usr   /fsys/allison/usr   nfs    defaults   0    0
```

Данная строка файла *fstab* напоминает записи, соответствующие описаниям монтирования локальных файловых систем, только в первой колонке указыва-

ется не имя устройства, а имя удаленной системы, а в колонке, где указывается тип файловой системы, следует указать тип `nfs`. В данном случае запись соответствует удаленному каталогу */usr* на машине `allison`, который на этапе загрузки системы будет монтироваться в каталог */fsys/allison/usr*.

Как и в случае монтирования обычных файловых систем, каталог точки монтирования (в данном случае */fsys/allison/usr*) должен быть создан заранее, до того как будет выполнена попытка смонтировать удаленный каталог. В данном примере строка в файле */etc/fstab* позволит монтировать удаленный каталог */usr*, расположенный на машине `allison`.

Операция монтирования может характеризоваться дополнительными параметрами. Наиболее часто используются параметры `soft` и `hard`. Параметр `soft` означает, что когда попытка обращения к удаленному файлу будет терпеть неудачу, клиент NFS будет отправлять вызывающему приложению сообщение об ошибке. Некоторые приложения достаточно грамотно обслуживают такие ситуации, некоторые – нет. Параметр `hard` означает, что клиент NFS будет «подвешивать» систему, если сервер прекратит откликаться на запросы. За дополнительной информацией по каждому из параметров обращайтесь к страницам справочного руководства команды `mount`.

Не забывайте проверять параметры `ro` и `rw`. Экспортируя каталог, администратор может предоставить доступ к нему *только для чтения*, в этом случае вы не сможете выполнять запись в смонтированную файловую систему. В такой ситуации в поле `options` файла */etc/fstab* следует указать параметр `ro` вместо `defaults`.

Чтобы проверить, был ли экспортирован требуемый каталог на удаленной системе (см. раздел «Экспортирование каталога на стороне сервера NFS» далее в этой главе), а заодно протестировать настройки локальной системы, можно выполнить следующую команду с привилегиями пользователя *root:*

```
# mount allison:/usr
```

Поиск экспортированных ресурсов NFS и диагностика неисправностей

Иногда бывает известно заранее, что на некотором сервере должен существовать определенный ресурс файловой системы NFS, но при этом неизвестно, был ли запущен сервер NFS или, возможно, не известно правильное название разделяемого ресурса. В следующем примере демонстрируется, как можно отыскать доступные ресурсы. В данном случае имеются три сервера NFS: *merlin*, *frodo* и *sunsol*. Посмотрим, какие ресурсы экспортируются каждым из них.

Утилита, которая обычно используется для проверки служб NFS, называется *showmount*. Как правило, этот инструмент доступен для использования только пользователю *root*. Проверка всех трех серверов может быть выполнена следующим образом:

```
linux:~ # showmount -e merlin
Export list for merlin:
/srv  *.myworld.org,192.168.1.0/24
/data *.myworld.org,192.168.1.0/24
```

Сервер *merlin* экспортирует два каталога. Доступ к ним возможен только для клиентов, расположенных в домене *myworld.org*, а также из сети с адресом *192.168.1.0*. Посмотрим, какие сюрпризы приготовлены на сервере *sunsol*:

```
linux:~ # showmount -e sunsol
Export list for sunsol:
/export (everyone)
```

Здесь каталог */export* экспортируется всему миру. Такой тип экспорта обычно используется для экспортирования ресурса в частной сети и никогда на машине, подключенной к Интернету. Заметим, что такой результат означает доступность экспортируемого каталога для монтирования на чтение любым желающим. Если же окажется, что в каталог можно еще и записывать, то это уже можно считать серьезной брешью в системе безопасности системы!

В заключение посмотрим, что произойдет, когда запрос информации об экспортируемых каталогах будет отправлен серверу, где служба NFS не была запущена. Отправим запрос серверу *frodo*, где служба NFS, возможно, не была настроена или не запущена по некоторым причинам:

```
linux:~ # showmount -e frodo
mount clntudp_create: RPC: Program not registered
```

Здесь видно, что процесс вызова удаленных процедур (Remote Procedure Call, RPC), через который выполняется доступ к ресурсам NFS, не был запущен. RPC – это протокол, обеспечивающий взаимодействие между клиентом и сервером. Проверить, какие службы RPC были запущены, можно с помощью утилиты `rpcinfo`. В данном примере мы попробуем найти все отличия между серверами `merlin` и `frodo`:

```
linux:~ # rpcinfo -p merlin
   program vers proto   port
    100000    2   tcp    111 portmapper
    100000    2   udp    111 portmapper
    100003    2   udp   2049 nfs
    100003    3   udp   2049 nfs
    100227    3   udp   2049 nfs_acl
    100003    2   tcp   2049 nfs
    100003    3   tcp   2049 nfs
    100227    3   tcp   2049 nfs_acl
    100024    1   udp   1254 status
    100021    1   udp   1254 nlockmgr
    100021    3   udp   1254 nlockmgr
    100021    4   udp   1254 nlockmgr
    100024    1   tcp   4777 status
    100021    1   tcp   4777 nlockmgr
    100021    3   tcp   4777 nlockmgr
    100021    4   tcp   4777 nlockmgr
    100005    1   udp    645 mountd
    100005    1   tcp    648 mountd
    100005    2   udp    645 mountd
    100005    2   tcp    648 mountd
    100005    3   udp    645 mountd
    100005    3   tcp    648 mountd
```

Служба RPC *nlockmgr* предоставляет возможность выполнения блокировок файлов через соединения, связанные со смонтированными ресурсами NFS, а служба RPC *nfs_acl* предоставляет возможность управления системой безопасности на базе списков управления доступом стандарта POSIX (POSIX Access Control List, ACL). Ниже приводится результат выполнения аналогичного запроса к серверу *frodo*.

```
linux:~ # rpcinfo -p frodo
   program vers proto   port
    100000    2   tcp    111 portmapper
    100000    2   udp    111 portmapper
    100024    1   udp  32768 status
    100021    1   udp  32768 nlockmgr
    100021    3   udp  32768 nlockmgr
    100021    4   udp  32768 nlockmgr
    100024    1   tcp  32768 status
    100021    1   tcp  32768 nlockmgr
    100021    3   tcp  32768 nlockmgr
    100021    4   tcp  32768 nlockmgr
```

Здесь служба RPC *nfs* отсутствует. Минуту спустя мы запустили ту же самую команду и получили следующий результат:

```
linux:~ # rpcinfo -p frodo
rpcinfo: can't contact portmapper: RPC: Remote system error - Connection refused
```

который говорит о том, что служба `portmapper`, обеспечивающая поддержку возможностей RPC, была остановлена. Вероятно, это было сделано с целью выполнить некоторые обслуживающие процедуры, или сервер находился на стадии остановки системы.

Теперь в ваших руках есть ключ, с помощью которого можно обнаружить и диагностировать доступность NFS, а также возможные причины неполадок, связанных с NFS.

Монтирование файловых систем NFS вручную

Определить, какие файловые системы NFS в настоящее время смонтированы в системе Linux, можно с помощью утилиты `mount`:

```
linux:~ # mount -t nfs
merlin:/data on /data type nfs (rw,addr=192.168.1.4)
merlin:/srv on /msrv type nfs (rw,addr=192.168.1.4)
sunsol:/export on /mnt type nfs (ro,addr=192.168.1.6)
```

Ресурсы NFS, экспортируемые сервером *merlin*, были смонтированы в режиме, который обеспечивает доступ на чтение-запись. Ресурс, экспортируемый сервером *sunsol*, был смонтирован в режиме только для чтения.

Предположим, что необходимо смонтировать ресурс */export/work*, экспортируемый сервером *sunsol*, в каталог */home/work* рабочей станции Linux, тогда сделать это можно будет с помощью следующих команд:

```
linux:~ # mkdir /home/work
linux:~ # mount sunsol:/export/work /home/work
```

Команда `df` поможет определить, какие ресурсы были смонтированы, а также покажет объем доступного дискового пространства:

```
linux:~ # df /home/work
Filesystem           1K-blocks      Used Available Use% Mounted on
sunsol:/export/work   17645600   3668352  13800800  21% /home/work
```

В случае необходимости размонтировать ресурс NFS можно обычной командой:

```
linux:~ # umount /home/work
```

Немного попрактиковавшись, вы быстро наберетесь опыта использования клиентских инструментальных средств NFS.

Экспортирование каталога на стороне сервера NFS

Как уже говорилось выше, мы не будем пытаться рассказывать о настройке сервера NFS, а лишь коротко поясним, как экспортировать каталог после того, как сервер будет запущен. В нашем примере системный администратор сервера *allison* должен настроить экспорт каталога (в данном случае */usr*), чтобы его можно было смонтировать на другой системе. В большинстве систем UNIX сделать это можно за счет простого редактирования файла */etc/exports*. Совершенно не обязательно, чтобы экспортируемый каталог являлся корневой файловой системой, то есть сервер может экспортировать каталог */usr*, даже если он не имеет своей собственной отдельной файловой системы.

Попробуем взять на себя обязанности системного администратора сервера NFS и экспортировать каталог */data/accounts*, чтобы его могли использовать все клиенты NFS из домена *myworld.org*. Сделать это можно, выполнив следующие шаги:

1. Создать каталог */data/accounts*:

   ```
   linux:~ # mkdir -p /data/accounts
   linux:~ # chmod 770 /data/accounts
   linux:~ # chown bill:accounting /data/accounts
   ```

   Первая команда создаст экспортируемый каталог (если он не существовал ранее), вторая настроит биты прав доступа так, чтобы каталог стал доступен членам группы, а третья установит разрешения чтения и записи для группы *accounting*. Кроме того, члены этой группы смогут получать список файлов, расположенных в этом каталоге.

2. Создать файл */etc/exports* со следующим содержимым:

   ```
   /data/accounts   *.myworld.org(rw,no_root_squash,sync)
   ```

   и установить владельца и права доступа к файлу:

   ```
   linux:~ # chown root:root /etc/exports
   linux:~ # chmod 644 /etc/exports
   ```

Настройка системы для работы в качестве клиента NIS

Информационная сетевая система NIS не является инструментом, имеющим непосредственное отношение к совместному использованию файлов и принтеров, но мы представим ее в этой главе, потому что данная система использует некото-

рые компоненты вместе с родственной ей NFS, а также потому, что ее использование может упростить работу с NFS, так как позволяет каждому пользователю иметь одинаковые числовые идентификаторы на всех системах.

NIS является сложной системой просто потому, что она очень гибкая. Это сетевая база данных общего назначения, дающая вашей системе прозрачный доступ к информации об учетных записях пользователей, группах, файловых системах и т. д. из баз данных, находящихся в сети.

Цель NIS состоит в облегчении управления сетями. Например, если информация об учетных записях пользователей (такая как в */etc/passwd*) хранится на одном сервере, этими записями можно пользоваться на нескольких машинах. В предыдущем разделе, посвященном NFS, мы показали, что для эффективного удаленного доступа к файлам UID и GID на клиентах и серверах NFS должны совпадать. Использование NIS позволяет определять *uid* и *gid* не локально, а удаленно.

Если в сети, куда подключена ваша машина, уже используется NIS, скорее всего, вы сможете добавить вашу машину в качестве клиента NIS, дав ей таким образом возможность напрямую получать из сети базы данных пользователей, групп и другие. До некоторой степени это вообще делает ненужным создание локальных учетных записей пользователей или групп. Кроме пользователей, создаваемых локально, таких как *root*, *bin* и т. д., все другие пользователи создаются на сервере NIS. Если совместить использование NIS с монтированием личных каталогов пользователей на сервере NFS, нет также необходимости выделять пользователям место на локальном диске. NIS может значительно сократить объем работы системного администратора.

В системах NIS могут быть *серверы*, *подчиненные серверы* и *клиенты*. Как вы могли догадаться, *серверы* – это системы, где создаются и откуда управляются базы NIS. *Подчиненные серверы* NIS – это системы, куда сервер копирует свои базы данных. Они могут предоставлять информацию другим системам, но изменения в самих базах должны выполняться на основном сервере. *Клиенты* NIS – это системы, запрашивающие информацию из баз данных серверов.

Чтобы полностью разобраться, как работает NIS и как управлять сервером NIS, потребовалось бы прочесть целую книгу (снова отсылаем к «Managing NFS and NIS»). Однако при чтении материала по NIS вы можете столкнуться с различными терминами. Первоначально NIS называлась «Yellow Pages», однако сейчас это название не используется, поскольку Yellow Pages является зарегистрированной в Великобритании торговой маркой (это телефонная книга, в конце концов), но наследие старого названия по-прежнему можно наблюдать в именах команд, содержащих символы *yp*.

Существуют, по крайней мере, две реализации NIS для Linux: «традиционная» реализация NIS и отдельная реализация, известная как NYS (сокращение от «NIS+», «YP» и «Switch»). Программный код клиента NIS для «традиционной» реализации содержится в стандартной библиотеке языка программирования C и уже установлен на большинстве систем Linux. (Это необходимо, чтобы такие программы, как `login`, могли иметь прозрачный доступ как к базам NIS, так и к локальным системным файлам.) Стандартная библиотека языка программирования C – *glibc2*, устанавливаемая в настоящее время большинством дистрибутивов, содержит поддержку NIS+. Код клиента NYS содержится в Network

Services Library (библиотека сетевых служб) – *libnsl*. В системах Linux, использующих NYS, такие программы, как `login`, должны быть скомпилированы с использованием этой библиотеки.

Различные дистрибутивы Linux используют разные версии клиентского программного кода NIS или NYS, а некоторые используют смесь из обоих. Для уверенности мы опишем, как настроить систему и для традиционной реализации NIS, и для NYS. Поэтому, что бы ни было установлено на вашей системе, она должна работать как клиент.

Еще более осложняет ситуацию система подключаемых модулей аутентификации PAM (Pluggable Authentication Modules), поставляемая в составе ряда дистрибутивов и упоминавшаяся в разделе «PAM и другие способы аутентификации» главы 11. В этом случае программы типа `login` компонуются с библиотекой PAM, которая, в свою очередь, загружает модуль библиотеки PAM, реализующий принятую в системе схему аутентификации или делегирующий задачу другим библиотекам.

Мы предполагаем, что администратор вашей локальной сети уже установил и запустил все необходимые демоны NIS (такие как *ypbind*), используемые традиционной реализацией NIS для связи с NIS-сервером. Если ваша Linux-система не поддерживает NIS, обратитесь к документации, например «Linux NIS HOWTO», чтобы настроить ее «с нуля». Практически все дистрибутивы Linux сейчас поставляются в комплекте с поддержкой NIS-клиента (и сервера), и все, что от вас потребуется, – это отредактировать несколько файлов с настройками.

Первый шаг заключается в установке домена NIS, в котором будет работать ваша система. Эту информацию вам могут предоставить сетевые администраторы. Обратите внимание, что имя домена NIS не обязательно должно совпадать с именем домена DNS, которое устанавливается командой `hostname`. Например, если полное имя системы – *loomer.vpizza.com*, то имя домена DNS будет *vpizza.com*. Однако имя домена NIS может быть абсолютно другим, например *vpizzas*. Имя домена NIS выбирается администратором сервера NIS и не связано с ранее описанным именем домена DNS.

Имя домена обычно устанавливается командой *domainname* во время загрузки, возможно, в одном из системных *rc*-файлов (например, в */etc/rc.d/rc.inet1*, описанном ранее). Сначала нужно убедиться, что *domainname* не выполняется в одном из существующих *rc*-файлов. Команда использует следующий формат:

```
linux:~ # domainname
domain-name
```

например `domainname vpizzas`. Сама команда обычно находится в */sbin/domainname* и может иметь слегка отличающееся имя, такое как *domainname-yp*.

Установка имени домена под NYS происходит несколько иначе. Вы должны создать (или отредактировать) файл */etc/yp.conf*. Этот файл должен содержать две строки: одну, указывающую имя вашего домена NIS, и другую, указывающую имя хоста сервера NIS. Например, строки

```
linux:~ # domain vpizzas
linux:~ # ypserver allison.vpizza.com
```

устанавливают имя домена NIS равным *vpizzas* и указывают, что в качестве сервера NIS следует использовать *allison.vpizza.com*. Если в этом файле нет строки `ypserver`, система рассылает широковещательное сообщение во время загрузки для определения имени сервера NIS. Имя хоста нужного сервера NIS вам могут предоставить сетевые администраторы.

После выполнения этих двух шагов ваша система должна получить возможность прозрачного доступа к базам NIS. Простым способом проверки может быть запрос к базе данных паролей на сервере NIS. Команда *ypwhich* выполняет запрос к специальным базам NIS, например:

```
linux:~ # ypwhich username passwd
```

Если будет возвращена строка из NIS-базы *passwd* для заданного пользователя, значит, запрос к базе NIS был выполнен успешно. (Можно проверить корректность возвращаемой информации, выполнив ту же команду с другой системы в вашем домене NIS, где настройка NIS гарантированно работает.) База данных NIS *passwd* не идентична файлу */etc/passwd* на вашей системе, хотя и использует тот же формат. Дополнительная информация по устранению неполадок в вашей конфигурации NIS содержится в документах Linux HOWTO.

16

X Window System

В главе 3 было дано краткое введение в графические окружения рабочих столов Linux, а в последующих главах рассказывалось об удобных и мощных инструментах, которые можно в них использовать. Довольно редко приходится иметь дело с настройками программного обеспечения, составляющего основу графического интерфейса, но иногда разрешение экрана может быть неудовлетворительным или вообще не удается запустить графическую оболочку. В таких ситуациях можно заметить в системных журналах массу сообщений об ошибках, имеющих отношение к X-серверу или к различным файлам и библиотекам, имена которых начинаются с символа `X`.

По сути, X Window System включает в себя все программное обеспечение, которое позволяет центральному процессору взаимодействовать с видеокартой и делает возможным отображение графики на экране монитора. Однако зона влияния X простирается намного дальше: эта система обеспечивает интерфейс практически неограниченной гибкости, позволяя программам отображать графику, взаимодействовать с пользователем и обмениваться данными с другими программами, имеющими графический интерфейс. KDE и GNOME представляют собой наборы библиотек и инструментальных средств, работающие под управлением X. В этой главе будет говориться о том, как установить и настроить X Window System в случае, если это не было сделано в процессе установки дистрибутива.

История развития X

Сложно описать X Window System в двух словах. X – это полнооконный графический интерфейс, который имеется практически во всех компьютерных системах[1],

[1] В составе операционных систем семейства Windows нет реализации системы X, да она и неорганична для этих систем, базирующихся на собственной графической подсистеме. Реализации X для Windows существуют только в виде продуктов сторонних производителей, предназначенных главным образом для совместимости с системами UNIX. Примером одной из наиболее развитых реализаций такого рода является система Exceed канадской фирмы Hummingbird Communication Ltd. – *Примеч. науч. ред.*

но в основном предназначался для UNIX, а теперь и для Linux. Он предоставляет огромное количество возможностей как для программиста, так и для пользователя. В частности, есть как минимум полдюжины *менеджеров окон* для X, каждый из которых предлагает свой интерфейс для управления окнами. Менеджер окон и окружение рабочего стола выбираются при установке дистрибутива. Настроив атрибуты менеджера окон, вы можете полностью контролировать положение окон на экране, их цвет и рамки, используемые для оформления, и т. д.

Система X первоначально была разработана в рамках проекта Athena в Массачусетском технологическом институте (MIT), Digital Equipment Corporation и IBM. На момент написания этих строк текущей версией X была версия 11 в редакции 6 (X11R6), которая впервые была представлена в апреле 1994 года, и последующие версии изменялись в младшем номере. После этого релиза X стала стандартом де-факто для графических сред UNIX-систем.

Параллельно с коммерческим использованием X Window System продолжает распространяться по свободной лицензии от Open Group. Поэтому полная реализация X свободно доступна для систем Linux. В Linux наиболее часто используется X.org – реализация X, основанная на исходных текстах X. Сейчас эта версия поддерживает не только системы Intel, но и платформы Alpha AXP, MicroSPARC, PowerPC и другие архитектуры. Планируется поддержка других архитектур. В X.org была добавлена поддержка огромного числа графических карт и множества других операционных систем (включая Linux). X.org представляет собой реализацию последней версии, X11R6.8.2.[1]

Следует отметить, что существуют коммерческие серверы X Window System для Linux, которые обладают некоторыми преимуществами перед X.org (такими как поддержка некоторых дополнительных видеокарт). Тем не менее большинством используется X.org, с которой и следует начинать.

Как уже упоминалось в разделе «Зачем нужен графический рабочий стол?» главы 3, многие используют Linux в качестве серверной платформы и вообще не устанавливают X-сервер. Администрирование сервера в таких случаях зачастую выполняется удаленно или исключительно с использованием интерфейса командной строки.

Основы X

X основана на модели «клиент-сервер», в которой *X-сервер* – это программа, работающая на вашей системе и обрабатывающая все обращения к графическому оборудованию. *X-клиент* – это прикладная программа, которая общается с сервером, посылая запросы типа «нарисовать линию» или «обратить внимание на ввод с клавиатуры». X-сервер стремится обслужить эти запросы, рисуя на экране линию или посылая пользовательский ввод (с клавиатуры, мыши или какого-нибудь другого устройства ввода) приложению-клиенту. Примерами X-клиен-

[1] X.org – относительно недавняя версия. Появившиеся разногласия в сообществе X Window System привели к расколу проекта. Многие перешли с ранее преобладавшей версии XFree86 на более новую версию X.org. Мы не будем комментировать эти разногласия, поскольку основная их причина лежит в плоскости личных взаимоотношений, а не в стремлении к техническим улучшениям.

тов могут служить ставший знаменитым графический редактор GIMP и многие программы, входящие в упомянутые графические среды KDE и GNOME, например почтовая программа KMail из KDE.

Важно отметить, что X – это графическая система, ориентированная на работу в сети, то есть клиенты X могут работать локально (на той же системе, где работает X-сервер) или удаленно (на системе, находящейся в сети TCP/IP). X-сервер прослушивает и локальные, и удаленные сетевые сокеты, ожидая поступления запросов от клиентов. Очевидно, что это весьма мощная возможность. Если есть подключение к сети TCP/IP, можно запустить приложение X на удаленной машине в сети и направить результаты его работы на монитор вашего локального X-сервера.

Другими преимуществами X являются безопасность (если пользователь пожелает), модульное разделение функций и поддержка множества различных архитектур. Все это делает X Window System технически более совершенной, чем все другие оконные системы.

X Window System делает различие между поведением приложения и *управлением окнами*. Клиенты, работающие под X, отображаются на экране в одном или нескольких *окнах*. Однако управление окнами (позиция на экране, изменение размеров и т. д.) и их оформление (вид оконных рамок) не контролируются X-сервером. Этим занимается другой клиент X, называемый *менеджером окон*, который работает одновременно с другими клиентами X. Выбор оконного менеджера в некоторой степени определяет, как будет выглядеть система X в целом. Большинство менеджеров окон обладают высокой гибкостью и множеством настроек. Путем редактирования файлов с настройками пользователь может выбрать внешний вид оформления окон, политику передачи фокуса ввода, назначение кнопок мыши в зависимости от того, где находится указатель – над рабочим столом или окном приложения, – и многое другое. Современные системы позволяют выполнять эти настройки даже через графический интерфейс.

Для того чтобы полностью понять концепцию менеджеров окон, нужно усвоить, что менеджер окон не влияет на то, что клиентское приложение выводит в свое окно. Он отвечает только за оформление окна, то есть за рамки и кнопки, позволяющие закрывать окна, передвигать их и изменять размеры.

В любом X-сервере может быть только один менеджер окон. Теоретически можно обойтись и без оконного менеджера, но тогда вы не сможете перемещать окна по экрану, поместить окно на передний план, минимизировать, разворачивать или изменять размеры окон, если сами программы не предоставят такую функциональность.

Коротко еще раз коснемся настольных графических сред. Графическая среда типа KDE или GNOME – это набор приложений и утилит со стандартизированным внешним видом и многими другими общими свойствами: например, меню всех приложений могут быть организованы по одному и тому же принципу. Графическим средам в X всегда нужен менеджер окон, как говорилось выше. Некоторые графические среды предоставляют собственный менеджер окон (например, KWin в среде KDE), в других собственного менеджера окон нет. Выбор менеджера окон остается за пользователем.

Требования к оборудованию

В этом разделе указаны чипсеты для видеоадаптеров, поддерживаемые в X.org версии 6.8.2, вышедшей в феврале 2005 года. В документации к вашему видеоадаптеру должен быть указан используемый чипсет. Если вы хотите приобрести новую видеокарту или новый компьютер, который поставляется с видеоадаптером, попросите поставщика уточнить изготовителя, модель и набор микросхем. Для этого может потребоваться обратиться в отдел технической поддержки продавца – как правило, продавцы не отказывают в этом. Многие продавцы оборудования для PC заявляют при этом, что видеокарта является «стандартной SVGA-картой» и «должна работать» в вашей операционной системе. Объясните им, что ваше программное обеспечение (Linux и X.org!) не поддерживает все типы видеочипсетов и вам требуется подробная информация.

Хорошим источником для определения поддержки вашей видеокарты X-сервером является сайт *http://www.x.org/X11R6.8.2/doc/RELNOTES3.html.#9*.

Если есть сомнения по поводу поддержки чипсета имеющейся у вас видеокарты, можно попробовать запустить команду:

```
Xorg -configure
```

Она проверит имеющееся в наличии оборудование и создаст начальный файл с настройками, который можно будет отредактировать в соответствии со своими желаниями.

Следует отметить, что проект X.org недавно перешел на совершенно новую архитектуру драйверов, которая отличается значительно большей гибкостью, чем прежняя, и позволит более оперативно осуществлять поддержку новой видеоаппаратуры.

Видеокарты с поддерживаемыми наборами микросхем обычно поддерживаются для всех типов шин, включая PCI и AGP.

Все эти чипсеты поддерживаются в режиме 256 цветов, некоторые из них поддерживаются в моно- и 16-цветном режиме и даже при большей глубине цвета.

Со временем этот список будет, несомненно, расширяться. Полный список поддерживаемых видеочипсетов должен находиться в замечаниях к текущей версии X.org. Кроме того, всегда просматривайте файл *README* для вашего конкретного чипсета.

Помимо этих наборов микросхем в ядро, начиная с серии 2.2, включена также поддержка устройства буфера кадров через драйвер `fbdev`. Если ваша видеокарта поддерживается любым обычным драйвером X-сервера, для достижения большей производительности следует использовать его, если нет, то возможно, что запустить X все же удастся с помощью буфера кадров. Для некоторого оборудования даже устройство буфера кадров обеспечивает поддержку графического ускорения.

Одной из проблем, с которой столкнулись разработчики X.org, стало использование производителями видеокарт нестандартных механизмов определения частот синхронизации для управления картой. Некоторые производители либо не описывали способов программирования карты, либо требовали подписания дополнительного соглашения о нераспространении полученной информации. Очевидно, что это ограничило бы свободное распространение X.org, чего, конечно, ко-

манда разработчиков X.org допускать не хочет. Таким образом, если какая-либо видеокарта не поддерживается, на это могут быть достаточно веские причины.

Определить минимальные требования к оборудованию, необходимые для нормальной работы X, довольно трудно, поскольку это зависит от большого числа внешних факторов, таких как количество программ с графическим интерфейсом, которые планируется запускать, набор функций, которые будет выполнять система, и т. д. Но любой компьютер, продававшийся в течение, скажем, последних 5–8 лет, должен работать вполне прилично, и, вероятно, многие более старые модели тоже. Поэтому, прежде чем делать решающий шаг и приобретать дорогостоящее оборудование, следует просмотреть документацию X и убедиться, что имеющаяся в наличии видеокарта поддерживается.

Сравнительные тесты производительности для различных видеокарт под X.org периодически появляются в телеконференциях Usenet *comp.windows.x.i386unix* и *comp.os.linux.misc*.

К слову, личная Linux-система одного из авторов (Калле) работает на платформе AMD K6-2 с 128 Мбайт оперативной памяти и видеокартой PCI Permedia II с 8 Мбайт DRAM. Такая конфигурация уже значительно превосходит по скорости работы видеоподсистемы многие рабочие станции. X.org на системе Linux с ускорителем SVGA выдаст большую производительность, чем можно встретить на коммерческих рабочих станциях UNIX (которые часто применяют обычные кадровые буферы для графики и предоставляют дорогие видеоускорители только как дорогостоящие расширения системы).

Вашей машине потребуется минимум 32 Мбайт физической оперативной памяти и 64 Мбайт виртуальной памяти (например, 32 Мбайт физической и 32 Мбайт подкачки). Помните, что чем больше у вас физической памяти, тем меньше система будет пользоваться подкачкой на диске при нехватке памяти. Поскольку свопинг всегда работает медленно (диски значительно медленнее, чем оперативная память), для комфортной работы с X.org необходимо 32 Мбайт и более оперативной памяти. Система с 32 Мбайт физической памяти может работать значительно медленнее (возможно, даже в 10 раз медленнее), чем система с 64 Мбайт или больше.

Установка X.org

Команда разработки X.org не выпускает дистрибутивов в двоичном виде, но в составе дистрибутива должна быть скомпилированная версия, и в большинстве случаев этого вполне достаточно. В случае необходимости полные исходные тексты X.org можно отыскать на сайте *ftp://ftp.x.org/pub/X11R6.8.2/src*, включая инструкции по сборке двоичных файлов. (Разумеется, к тому моменту, как эта книга попадет к вам в руки, номер последней версии может измениться.)

Создание файла с настройками X (с именем *XF86Config-4* или *xorg.conf*, в зависимости от версии и дистрибутива) с самого начала – занятие довольно утомительное, и потому не рекомендуется. В этом разделе будут описаны три способа создания, по крайней мере, начального файла с настройками. Используя сведения из этой главы, вы сможете внести в файл необходимые изменения, чтобы оптимизировать его под свою систему.

Первое, что можно попробовать (разумеется, после того, как была произведена попытка выполнить настройки в процессе установки дистрибутива), – это запустить программу с именем *xorgcfg*, которая поставляется вместе с X.org. Это программа установки графического окружения, которая работает даже в текстовом терминале, что позволяет пользоваться ею, даже когда система X Window еще не настроена.

Если *xorgcfg* не запускается, следующий шаг – попробовать воспользоваться командой, которая уже упоминалась выше, – *Xorg -configure*. Эта команда запускает X-сервер в специальном режиме, в котором будет выполнена попытка максимально точно определить тип имеющегося оборудования и создать скелет файла с настройками. Этого файла может оказаться вполне достаточно, чтобы запустить X-сервер, хотя может сохраниться потребность выполнить дополнительные настройки.

Если и команда *X -configure* потерпит неудачу (что весьма маловероятно), тогда можно обратиться к другой программе как к инструменту последней инстанции. Называется эта программа *xorgconfig* и должна поставляться вместе с X.org. В ходе исполнения она задаст ряд вопросов, касающихся оборудования. Если на некоторые из вопросов ответить окажется затруднительно, просто выбирайте значения по умолчанию и посмотрите, к чему это приведет. По окончании вы опять должны получить скелет файла с настройками.

Настройка X.org

Настроить X.org в большинстве случаев несложно. Однако, если драйвер для вашего оборудования находится в процессе разработки либо вы хотите добиться лучшей производительности или разрешения от видеокарты с графическим ускорителем, настройка X.org может потребовать некоторого времени.

В этом разделе мы опишем, как создать и отредактировать файл *xorg.conf*, который настраивает сервер X.org. Этот файл по умолчанию находится в каталоге */etc/X11/*, но поиск его ведется в самых разных местах, таким образом, ваш дистрибутив мог поместить его куда-нибудь в другое место. В большинстве случаев лучше всего начать со стартового файла настроек, созданного одним из способов, описанных выше. Затем перейти к низкому разрешению: хорошим выбором будет разрешение 640×480, поддерживаемое всеми видеокартами и мониторами. Когда X.org уже работает, используя стандартное низкое разрешение, вы можете затем подправить настройки для использования всех возможностей, предоставляемых видеокартой. Смысл такого подхода заключается в том, чтобы убедиться в работоспособности X.org на вашей системе и в успешной установке, прежде чем пытаться настроить X.org для реального использования (иногда это бывает сложно). С оборудованием, используемым в настоящее время, вы можете легко получить разрешение до 1280×1024 пиксела (1024×768 – для большинства ноутбуков).

В дополнение к информации, которая приводится здесь, желательно ознакомиться с документацией на сайте *http://www.x.org/X11R6.8.2/doc/* и особенно с файлами *README*, имеющими отношение к вашей видеокарте.

Основным файлом с настройками, который необходимо создать, является */etc/X11/xorg.conf*. Этот файл содержит информацию о мыши, параметрах видеокарты и т. п. В качестве примера можно использовать файл */etc/X11/xorg.conf.install*,

поставляемый в составе дистрибутива X.org. Скопируйте этот файл в *xorg.conf* и начните его редактировать, если ни один из предложенных ранее методов создания начального файла не дал положительных результатов.

Формат файла *xorg.conf* достаточно подробно описывается на страницах справочного руководства. Прочтите сейчас это руководство, если вы еще не сделали этого.

Сейчас мы рассмотрим пример файла *xorg.conf*, участок за участком. Этот файл может выглядеть не совсем так, как файл, входящий в дистрибутив X.org, но их структура совпадает. Формат файла *xorg.conf* может изменяться с каждой версией X.org, данная информация относится только к X.org версии 6.8.2.

Независимо от того, что вы собираетесь делать, не следует просто копировать файл с настройками, приведенный здесь, и пытаться использовать его в своей системе. Попытка использования файла с настройками, не соответствующими оборудованию, может заставить монитор работать со слишком высокой для него частотой; были сообщения о выходе из строя мониторов (особенно мониторов с фиксированной частотой[1]) при использовании неверно сконфигурированного файла X.org. Вывод отсюда один: следует быть абсолютно уверенным в том, что файл *xorg.conf* соответствует установленному оборудованию, прежде чем пытаться использовать его.

После сделанного предупреждения хотим отметить, что настройка X.org стала значительно менее опасной, чем несколько лет назад, поскольку теперь X-сервер очень хорошо обнаруживает недопустимые настройки.

Каждый раздел (section) файла *xorg.conf* начинается со строки `Section "имя_раздела"` и заканчивается строкой `EndSection`. Первая часть файла *xorg.conf* называется `Files` и выглядит следующим образом:

```
Section "Files"
  FontPath     "/usr/X11R6/lib/X11/fonts/misc:unscaled"
  FontPath     "/usr/X11R6/lib/X11/fonts/local"
  FontPath     "/usr/X11R6/lib/X11/fonts/75dpi:unscaled"
  FontPath     "/usr/X11R6/lib/X11/fonts/100dpi:unscaled"
  FontPath     "/usr/X11R6/lib/X11/fonts/Type1"
  FontPath     "/usr/X11R6/lib/X11/fonts/URW"
  FontPath     "/usr/X11R6/lib/X11/fonts/Speedo"
  FontPath     "/usr/X11R6/lib/X11/fonts/PEX"
  FontPath     "/usr/X11R6/lib/X11/fonts/cyrillic"
  FontPath     "/usr/X11R6/lib/X11/fonts/latin2/misc:unscaled"
  FontPath     "/usr/X11R6/lib/X11/fonts/latin2/75dpi:unscaled"
  FontPath     "/usr/X11R6/lib/X11/fonts/latin2/100dpi:unscaled"
  FontPath     "/usr/X11R6/lib/X11/fonts/latin2/Type1"
  FontPath     "/usr/X11R6/lib/X11/fonts/latin7/75dpi:unscaled"
  FontPath     "/usr/X11R6/lib/X11/fonts/baekmuk:unscaled"
  FontPath     "/usr/X11R6/lib/X11/fonts/japanese:unscaled"
  FontPath     "/usr/X11R6/lib/X11/fonts/kwintv"
  FontPath     "/usr/X11R6/lib/X11/fonts/truetype"
```

[1] Это могло иметь место только для очень старых моделей мониторов, которые на сегодня отсутствуют не только в продаже, но и в эксплуатации. – *Примеч. науч. ред.*

```
    FontPath     "/usr/X11R6/lib/X11/fonts/uni:unscaled"
    FontPath     "/usr/X11R6/lib/X11/fonts/CID"
    FontPath     "/usr/X11R6/lib/X11/fonts/ucs/misc:unscaled"
    FontPath     "/usr/X11R6/lib/X11/fonts/ucs/75dpi:unscaled"
    FontPath     "/usr/X11R6/lib/X11/fonts/ucs/100dpi:unscaled"
    FontPath     "/usr/X11R6/lib/X11/fonts/hellas/misc:unscaled"
    FontPath     "/usr/X11R6/lib/X11/fonts/hellas/75dpi:unscaled"
    FontPath     "/usr/X11R6/lib/X11/fonts/hellas/100dpi:unscaled"
    FontPath     "/usr/X11R6/lib/X11/fonts/hellas/Type1"
    FontPath     "/usr/X11R6/lib/X11/fonts/misc/sgi:unscaled"
    FontPath     "/usr/X11R6/lib/X11/fonts/xtest"
    FontPath     "/opt/kde3/share/fonts"
    InputDevices "/dev/ttyS0"
    InputDevices "/dev/ttyS1"
    InputDevices "/dev/ttyS2"
    InputDevices "/dev/ttyS3"
    InputDevices "/dev/ttyS4"
    InputDevices "/dev/ttyS5"
    InputDevices "/dev/ttyS6"
    InputDevices "/dev/ttyS7"
    InputDevices "/dev/ttyS8"
    InputDevices "/dev/psaux"
    InputDevices "/dev/logibm"
    InputDevices "/dev/sunmouse"
    InputDevices "/dev/atibm"
    InputDevices "/dev/amigamouse"
    InputDevices "/dev/atarimouse"
    InputDevices "/dev/inportbm"
    InputDevices "/dev/gpmdata"
    InputDevices "/dev/mouse"
    InputDevices "/dev/usbmouse"
    InputDevices "/dev/adbmouse"
    InputDevices "/dev/input/mice"
    InputDevices "/dev/input/event0"
    InputDevices "/dev/pointer0"
    InputDevices "/dev/pointer1"
    InputDevices "/dev/pointer2"
    InputDevices "/dev/pointer3"
EndSection
```

Таких строк может быть много. Каждая строка `FontPath` определяет путь к каталогу, содержащему шрифты X11. Обычно не следует изменять эти строки; необходимо только убедиться, что для каждого установленного типа шрифта есть запись `FontPath` (то есть для каждого подкаталога в каталоге */usr/X11R6/lib/X11/fonts*). Если в `FontPath` добавить строку `:unscaled`, шрифты из этого каталога не будут масштабироваться. Это часто полезно, так как значительно увеличенные шрифты выглядят ужасно. В этот раздел помимо `FontPath` можно добавить `RgbPath`, установив тем самым путь к базе данных цветов RGB (хотя необходимость в этом вряд ли появится), и `ModulePath` для указания на каталог с динамически подгружаемыми модулями. Эти модули в настоящее время используются для некоторых специальных устройств ввода, а также для расширений PEX и XIE.

Следующий раздел – `ServerFlags`, который определяет несколько глобальных параметров для сервера. Как правило, этот раздел пустой или очень короткий:

```
Section "ServerFlags"
  Option       "AllowMouseOpenFail"
EndSection
```

Здесь сказано, что мы хотим, чтобы X-сервер запускался даже тогда, когда он не может обнаружить мышь. Дополнительные параметры можно найти в документации на сайте *http://www.x.org*. Часто параметры автоматически обнаруживаются при запуске сервера, и их не нужно здесь перечислять.

Далее следует раздел `Module`, служащий для динамической загрузки дополнительных модулей X-сервера, таких как поддержка специальных устройств или графических библиотек вроде PEX. Этот раздел может также использоваться для загрузки библиотек, обеспечивающих поддержку `freetype`, видео и ускорителей трехмерной графики. Ниже приводится пример раздела `Module`:

```
Section "Module"
  Load         "v4l"
  Load         "extmod"
  Load         "type1"
  Load         "freetype"
  Load         "dbe"
  Load         "dri"
  Load         "speedo"
  Load         "glx"
EndSection
```

Следующие разделы имеют название `InputDevice`. Обычно их не менее двух: один для клавиатуры и один для мыши. Если есть другие устройства, например графический планшет, для них создаются отдельные разделы:

```
Section "InputDevice"
  Driver     "kbd"
  Identifier "Keyboard[0]"
  Option     "Protocol" "Standard"
  Option     "XkbLayout" "us,ru(winkeys)"
  Option     "XkbOptions" "grp:ctrl_shift_toggle,grp_led:scroll"
  Option     "XkbModel" "pc105"
  Option     "XkbRules" "xfree86"
EndSection

Section "InputDevice"
  Driver     "mouse"
  Identifier "Mouse[1]"
  Option     "ButtonNumber" "7"
  Option     "Device" "/dev/mouse"
  Option     "Name" "Autodetection"
  Option     "Protocol" "ExplorerPS/2"
  Option     "Vendor" "Sysp"
  Option     "ZAxisMapping" "4 5"
EndSection
```

Существуют и другие параметры настройки. Здесь приведены настройки клавиатуры для США с поддержкой ввода кириллических символов (переключение раскладок производится нажатием комбинации Ctrl+Shift). Если у вас другая клавиатура, измените соответствующие строки.

В разделе `mouse` X-серверу сообщается, каким образом подключена мышь (в данном случае это */dev/mouse*, что обычно является ссылкой на соответствующий последовательный порт, например */dev/ttyS0*), какого типа эта мышь (параметр `"Protocol"`) и некоторые другие детали. Важно правильно указать протокол, но упоминавшиеся программы конфигурирования обычно автоматически определяют протокол.

Для шинной мыши Logitech нужно указать `BusMouse`. Если мыши Logitech не являются шинными, то для старых мышей нужно задавать `Logitech`, а для более новых нужно указывать протокол `Microsoft` или `Mouseman`. Это пример того, что протокол не обязательно имеет какое-то отношение к названию изготовителя мыши.

Если у вас современная мышь последовательного порта, можно попробовать задать `Auto` для автоматического выбора драйвера мыши.

Если X-сервер запущен, легко проверить, правильно ли выбран драйвер мыши: при перемещении мыши ее указатель на экране должен совершать соответствующее перемещение. Если это так, то настройки, скорее всего, корректные. Если нет – попробуйте другой драйвер и проверьте, правильно ли указано устройство.

Следующий раздел файла *xorg.conf* – `Device`. Здесь указываются параметры видеокарты. Если видеокарт несколько, должно быть несколько разделов `Device`.

```
Section "Device"
  BoardName  "Radeon LW"
  BusID      "1:0:0"
  Driver     "radeon"
  Identifier "Device[0]"
  Screen     0
  Option     "Rotate" "off"
  VendorName "ATI"
EndSection
```

Первая строка, `BoardName`, служит просто напоминанием о том, какая карта здесь настраивается (существенно, если карт несколько!). Аналогично `VendorName` – строка в свободном формате, служащая чисто описательным целям. Даже строка `Identifier` может быть выбрана произвольно, но ей должна соответствовать строка `Device` в последующих разделах файла с настройками. Обычно здесь используют такие названия, как `Device [0]`, `Device [1]` и т. д.

`BusID` указывает фактическую графическую карту в терминах встроенной аппаратуры на шине PCI. `PCI 1:0:0` или, короче, `1:0:0` в большинстве случаев будет правильным значением, если вариант только один. Если есть сомнения, запустите X-сервер командой:

```
X.org -scanpci
```

и внимательно изучите полученный результат; в нем должна быть хотя бы одна видеокарта (возможно, среди других устройств, которые в данный момент нас не интересуют). Например, строка:

```
(1:0:0) Matrox unknown card (0x19d8) using a Matrox MGA G400 AGP
```

сообщает о наличии карты Matrox MGA G400 с установленным соединением AGP. Первые цифры в скобках указывают идентификатор шины PCI, о котором говорилось выше.

Раздел `Screen` обязателен для видеокарт, допускающих подключение нескольких мониторов; для прочих здесь всегда указывается `0`.

Строка `Driver` очень важна, поскольку определяет фактический графический драйвер, который должен быть загружен X-сервером. Узнать название правильного драйвера можно с помощью описывавшихся выше программ настройки или из результатов, которые дает запуск X-сервера командой:

```
Xorg -probeonly
```

Здесь можно выяснить, какую информацию X-сервер собрал о вашей аппаратуре, включая драйвер, который, по его мнению, должен быть использован.

В этом файле можно задать массу других параметров, включая набор микросхем видеокарты, RAMDAC и другие свойства аппаратуры, но X-сервер неплохо определяет их сам, поэтому обычно делать это не требуется. Если вы все же хотите задать параметры сами, прочтите файл *README* для конкретного драйвера, в котором должны быть перечислены параметры этого драйвера и их возможные значения.

Следующий раздел, `Monitor`, определяет характеристики вашего монитора. Как и в случае с другими разделами, в файле *xorg.conf* может быть более одного раздела `Monitor`. Это бывает необходимо, когда к вашей системе подключено несколько мониторов или один и тот же файл *xorg.conf* используется с несколькими конфигурациями оборудования. Однако обычно необходим только один раздел `Monitor`:

```
Section "Monitor"
  Option       "CalcAlgorithm" "CheckDesktopGeometry"
  DisplaySize  320 240
  HorizSync    28-60
  Identifier   "Monitor[0]"
  ModelName    "THINKPAD 1400X1050 LCD PANEL"
  Option       "DPMS"
  VendorName   "IBM"
  VertRefresh  50-60
  UseModes     "Modes[0]"
EndSection
```

Строка `Identifier` используется для задания произвольного имени записи `Monitor`. Это может быть любая строка; она используется потом для ссылки на раздел `Monitor` в файле *xorg.conf*.

`HorizSync` указывает возможные частоты горизонтальной развертки монитора в килогерцах. Если у вас многочастотный монитор (multisync), можно указать интервал значений (или несколько интервалов через запятую). Для мониторов с фиксированными частотами указывается список фиксированных значений, например:

```
HorizSync 31.5, 35.2, 37.9, 35.5, 48.95
```

В руководстве по эксплуатации монитора эти значения должны быть описаны в разделе технических спецификаций. Если у вас нет этой информации, следует получить ее у производителя или продавца вашего монитора. Кроме этих источников информации имеются и другие; они будут перечислены позже.

Следует проявлять осторожность, устанавливая эти значения. Значения `VertRefresh` и `HorizSync` (они описаны ниже) помогают убедиться, что ваш монитор не

будет поврежден неправильными настройками. Если они указаны неправильно, вы вряд ли будете довольны вашей установкой X. Возможны нестабильное изображение, мерцание или обычные видеопомехи типа «снега».

`VertRefresh` указывает допустимые частоты вертикальной развертки (или частоты вертикальной синхронизации) монитора в килогерцах. Как и в случае с `HorizSync`, это может быть диапазон значений или список фиксированных значений. Допустимые значения должны быть указаны в руководстве по эксплуатации монитора.

Параметры `HorizSync` и `VertRefresh` используются только для дополнительного контроля за тем, что разрешения монитора, которые вы указали, находятся в допустимых диапазонах. Это уменьшает риск повреждения монитора при попытке его использования на частотах, для которых он не предназначен.

Директивы `Modeline` и `Mode` используются, чтобы задать режимы разрешения монитора. В отличие от более старых версий X.org, они не являются обязательными. В приведенном разделе `Monitor`, взятом с ноутбука, их нет. Эта информация переместилась в следующий раздел – `Modes`.

Раздел `Modes`, создаваемый отдельно для каждого настраиваемого монитора, перечисляет различные видеорежимы, которые должен поддерживать X-сервер, например:

```
Section "Modes"
  Identifier "Modes[0]"
  Modeline     "800x600" 36.88 800 832 912 1024 600 601 604 621
  Modeline     "800x600" 40.00 800 840 968 1056 600 601 605 628 +HSync +VSync
  Modeline     "1400x1050" 109.01 1400 1480 1632 1864 1050 1051 1054 1083
  Modeline     "1280x1024" 98.60 1280 1352 1488 1696 1024 1025 1028 1057
  Modeline     "1280x960" 97.68 1280 1352 1488 1696 960 961 964 993
  Modeline     "1152x864" 78.82 1152 1216 1336 1520 864 865 868 894
  Modeline     "1024x768" 61.89 1024 1080 1184 1344 768 769 772 794
  Modeline     "800x600" 36.88 800 832 912 1024 600 601 604 621
  Modeline     "800x600" 40.00 800 840 968 1056 600 601 605 628 +HSync +VSync
  Modeline     "640x480" 23.06 640 656 720 800 480 481 484 497
  Modeline     "1400x1050" 109.01 1400 1480 1632 1864 1050 1051 1054 1083
EndSection
```

Строка `Identifier` ссылается на имя, указанное в разделе `Monitor`. Каждая из строк `Modeline` задает отдельный видеорежим. Формат `Modeline` следующий:

```
Modeline name dot-clock horiz-values vert-values
```

Здесь *`name`* – это произвольная строка, которую в дальнейшем можно использовать для ссылки на режим разрешения в этом файле, *`dot-clock`* – частота генератора, или *dot clock*, связанная с режимом разрешения. Обычно она указывается в мегагерцах и определяет скорость, с которой видеокарта должна посылать пикселы на монитор при данном разрешении. Группы значений *`horiz-values`* и *`vert-values`*, состоящие из четырех чисел каждая, определяют, когда электронная пушка должна включаться, и когда должны проходить импульсы горизонтальной и вертикальной синхронизации во время развертки луча.

Как выяснить значения строки `Modeline` для вашего монитора? Это нелегко, особенно потому, что значительная часть прежней документации по X.org более не поставляется, возможно, потому, что она устарела и пока еще не обновлена. Луч-

ше всего воспользоваться одной из программ, генерирующих файлы с настройками и описанных в предыдущем разделе, чтобы получить первоначальный набор значений и корректировать их, пока настройки не станут удовлетворительными. Например, если во время работы X.org изображение на экране несколько смещено или мерцает, нужно понемногу изменять значения, стараясь исправить картинку. Что именно нужно исправлять, сказать трудно, поскольку это зависит от графического устройства, но после некоторых экспериментов обычно можно добиться хороших результатов. Кроме того, нужно не забывать про ручки управления на самом мониторе! Часто необходимо изменить горизонтальный или вертикальный размер изображения после запуска X.org, чтобы изображение находилось в центре и имело правильный размер. Другой вариант – воспользоваться программой *xvidtune* (за информацией о порядке пользования программой обращайтесь к страницам справочного руководства), которая помогает получить все числа, необходимые для `Modeline`, опробовать их и даже отменить, если что-то сделано ошибочно.

Кроме того, в X.org входят так называемые стандартные режимы VESA, благодаря которым можно обойтись вообще без раздела `Modes`. Синхронизация VESA дает стандартные значения для `Modeline`, которые поддерживаются большинством мониторов, хотя и не позволяют полностью использовать возможности конкретных устройств.

Обратите внимание, что аргумент *name* в `Modeline` (в данном случае `"640x480"`) – это произвольная строка. Принято называть режим по значению разрешения экрана, но можно использовать любое название, которое описывает для вас разрешение монитора.

Для каждой строки `Modeline` сервер проверяет, попадают ли указанные для режима значения в интервал значений, определенных с помощью `HorizSync` и `VertRefresh`. Если они не совпадают, сервер выдает предупреждение при попытке запустить X (подробнее об этом позже).

Не следует пробовать использовать параметры синхронизации и другие значения `Modeline`, предназначенные для других мониторов, отличающихся от вашей модели. Если вы попытаетесь запустить монитор на частоте, для которой он не предназначен, то можно повредить его или полностью вывести из строя.

Следующий раздел файла называется `Screen` и задает комбинации мониторов и видеокарт, используемые с конкретным сервером:

```
Section "Screen"
  DefaultDepth 24
  SubSection "Display"
    Depth      15
    Modes      "800x600"
  EndSubSection
  SubSection "Display"
    Depth      16
    Modes      "1400x1050"
  EndSubSection
  SubSection "Display"
    Depth      24
    Modes      "1400x1050" "1280x1024" "1280x960" "1152x864" "1024x768"
               "800x600" "640x480"
  EndSubSection
```

```
  SubSection "Display"
    Depth      32
    Modes      "800x600"
  EndSubSection
  SubSection "Display"
    Depth      8
    Modes      "800x600"
  EndSubSection
  Device       "Device[0]"
  Identifier   "Screen[0]"
  Monitor      "Monitor[0]"
EndSection
```

Этот раздел связывает вместе определения устройств, экранов и мониторов и указывает глубину цвета, используемую с разными видеорежимами.

Наконец, раздел `ServerLayout` завершает картину, определяя одну активную конфигурацию, которая состоит из одного или нескольких разделов `Screen` и одного или нескольких разделов `InputDevice`. Если у вас система с несколькими мониторами (multihead system), в которой к каждой графической карте подключен монитор, либо одна графическая карта, допускающая подключение нескольких мониторов, то в этом разделе также задается их относительное расположение. Например:

```
Section "ServerLayout"
  Identifier   "Layout[all]"
  InputDevice  "Keyboard[0]" "CoreKeyboard"
  InputDevice  "Mouse[1]" "CorePointer"
  InputDevice  "Mouse[3]" "SendCoreEvents"
  Option       "Clone" "off"
  Option       "Xinerama" "off"
  Screen       "Screen[0]"
EndSection
```

Существуют и другие разделы, но они совершенно не обязательны; X-сервер может работать и без них.

Запуск X

Когда файл *xorg.conf* будет готов, можно запускать X-сервер. Сначала убедитесь, что каталог */usr/X11R6/bin* входит в список каталогов поиска.

Запуск X-сервера производится командой:

```
startx
```

Это внешний интерфейс к команде `xinit` (в случае, если вы привыкли использовать `xinit` на других UNIX-системах). Вы можете по-прежнему использовать `xinit`, предоставляющую полный контроль над запуском программы, но вам придется запускать все необходимые программы вручную.

Эта команда запускает X-сервер и выполняет команды, найденные в файле *.xinitrc* в домашнем каталоге пользователя. Файл *.xinitrc* – это лишь сценарий командной оболочки, содержащий команды запуска клиентов X-сервера. Если этот файл отсутствует, по умолчанию выполняется системный файл */usr/X11R6/lib/*

X11/xinit/xinitrc. Задав в домашнем каталоге другой файл *.xinitrc*, можно изменить параметры начального отображения, устанавливаемые при запуске X Window System.

Возможные проблемы

Часто при начальной загрузке X-сервера что-нибудь работает не совсем правильно. Практически всегда это связано с ошибками в файле *xorg.conf*. Как правило, неверно указаны значения синхронизации монитора или частоты видеокарты. Если изображение на экране закручено или его края размыты, возможно, что эти значения установлены неверно. Проверьте также, правильно ли указаны чипсет видеокарты и другие параметры в разделе `Device` файла *xorg.conf*. В современной версии есть только один исполняемый файл сервера, который загружает модуль, необходимый для конкретной видеокарты. То, какой модуль загружается, зависит от настроек в разделе `Device`.

Если ничто другое не помогает, попробуйте запустить «чистый» X-сервер:

```
Xorg > /tmp/x.out 2>&1
```

Вы можете затем завершить работу X-сервера (комбинацией клавиш Ctrl+Alt+Backspace) и проверить содержимое файла */tmp/x.out*. Сервер запишет туда все предупреждения и ошибки, например, о том, что видеокарта не поддерживает необходимую для монитора частоту. */tmp/x.out* может быть весьма полезен для диагностики проблем любого рода. Внимательно изучите его, если X-сервер не запускается вовсе, не обеспечивает нужные вам разрешения либо показывает неустойчивую, со «снегом» или другими помехами, не удовлетворяющую вас картинку. И даже если все работает нормально, вы можете заглянуть в этот файл, чтобы выяснить, как X-сервер распознал ваше оборудование. Строки, начинающиеся с символов `(**)`, содержат данные, введенные вами в файл конфигурации, а строки, начинающиеся с символов `(--)`, содержат данные, которые X-сервер выяснил самостоятельно.

Не забывайте, что можно использовать комбинацию Ctrl+Alt с клавишами «плюс» и «минус» на цифровой клавиатуре для переключения между режимами разрешения монитора, перечисленными в строке `Modes` в разделе `Screen` файла *xorg.conf*. Если режим с высоким разрешением работает некорректно, попробуйте перейти на меньшее разрешение. По крайней мере вы выясните, что настройки для низких разрешений работают правильно.

Попробуйте также использовать кнопки управления на мониторе. Часто эти настройки необходимо подкорректировать при запуске X, например, если изображение съезжает в одну сторону, обычно такое смещение можно исправить ручками управления горизонтальным и вертикальным смещением.

Обсуждению вопросов по XFree86 посвящена телеконференция *comp.windows.x.i386unix*. Просматривая сообщения, относящиеся к такой же конфигурации видеосистемы, как у вас, вы, возможно, обнаружите, что кто-то уже сталкивался с теми же проблемами. Если это не помогает, свяжитесь с вашим дистрибьютором Linux; возможно, его служба технической поддержки вам поможет.

Наконец X-сервер работает. Теперь можно вернуться к главе 3 и почитать об окружениях рабочего стола, которые работают поверх X-сервера. Хотя по-прежне-

му существует возможность использовать X-сервер в «чистом» виде, без окружения рабочего стола, и иметь всего несколько открытых окон на экране, но ради этого не стоило проходить через все перипетии установки и настройки X-сервера. К тому же современные окружения рабочего стола обладают такой гибкостью, что позволяют настраивать их по своему вкусу до самых тонких нюансов.

X и 3D

Конечно же, Linux обладает возможностью отображать в своей графической среде не только двумерные окна и структуры, но и трехмерные графические изображения. Существует технология, ставшая стандартом де-факто программирования трехмерной графики, – это OpenGL, первоначально использовавшаяся на больших и мощных рабочих станциях, работавших под управлением UNIX. Ныне эта технология прекрасно поддерживается операционной системой Linux на компьютерах, оснащенных недорогими видеокартами, доступных для PC. В этом разделе мы рассмотрим, как настроить возможность отображения трехмерной графики.

Установка OpenGL

Как и в случае большинства других подсистем свободно распространяемой операционной системы Linux, у нас имеется достаточно широкий выбор реализаций OpenGL. Среди них можно назвать такие реализации, как Mesa, TinyGL и YGL. Самой заметной и к тому же ставшей стандартом де-факто для Linux является реализация Mesa.

GLX

Сама по себе технология OpenGL является нейтральной по отношению к платформе, таким образом, чтобы «приклеить» OpenGL к определенной оконной системе, требуется некоторое расширение. Для X11 таким расширением является GLX. GLX содержит расширения протокола X, которые позволяют отправлять команды рисования графики OpenGL через X-сокеты в X-сервер. Такой прием называют *косвенной визуализацией* (*indirect rendering*). В X.org имеется еще одна возможность, которая намного быстрее, но работает только для локального дисплея. Эту возможность называют *прямой визуализацией* (*indirect rendering*), она описывается в следующем разделе.

DRI

Начиная с версии 4 и выше, X.org включает в себя платформу, предоставляющую возможность прямого и эффективного доступа к видеокарте. Называется эта платформа инфраструктурой прямой визуализации (Direct Rendering Infrastructure, DRI). Она ускоряет работу реализации OpenGL, расположенной поверх нее. Платформа DRI состоит из нескольких компонентов:

- Модуль ядра, выполняющий мультиплексирование графического аппаратного обеспечения, чтобы оно могло использоваться несколькими процессами. Этот модуль называется менеджером прямой визуализации (Direct Rendering Manager, DRM) и предназначен для управления конкретной видеокартой. Обычно такие модули располагаются в каталоге */lib/modules/2.x.y/kernel/drivers/char/drm*. Требуемый модуль загружается ядром автоматически во время запуска X-сервера.

- Драйвер двумерной графики X.org. Для каждого типа видеокарт в X.org существует свой драйвер двумерной графики, который инициализирует дисплей, выполняет рисование двумерных объектов и т. д. Обычно эти драйверы находятся в каталоге */usr/X11R6/lib/modules/drivers/*.
- Драйвер трехмерной графики DRI. Этот компонент взаимодействует с той частью видеокарты, которая отвечает за вывод трехмерной графики, и эффективно преобразует команды OpenGL в команды, которые будут понятны видеоаппаратуре. Когда используется режим прямой визуализации, драйвер DRI загружается библиотекой *libGL.so*, благодаря чему приложение может обращаться к видеокарте напрямую, минуя X-сервер. Драйверы DRI обычно располагаются в каталоге */usr/X11R6/lib/modules/dri*.
- *libGL* – библиотека OpenGL, с которой должны быть связаны приложения, чтобы иметь возможность использовать OpenGL. При работе в режиме прямой визуализации библиотека *libGL* загружает драйвер DRI и работает непосредственно с ним, а при работе в режиме косвенной визуализации (например, при выводе на удаленный дисплей X) она создает команды GLX, которые посылаются X-серверу через обычные X-сокеты.

Закрытые драйверы

К сожалению, не все изготовители видеокарт стремятся передать в публичное пользование информацию о том, как работают их видеокарты. Особенно это относится к современным аппаратным ускорителям трехмерной графики. Но, к счастью, архитектура драйверов XAA, реализованная в X.org, совместима на уровне двоичных программных кодов даже с версиями драйверов, разработанными для других операционных систем (пока архитектура самих видеокарт остается неизменной), что позволяет устанавливать закрытые драйверы, содержащие только двоичные файлы.

В наши дни широкое распространение получили графические карты NVIDIA и ATI. Трехмерные возможности самых последних версий видеокарт этих производителей пока еще не поддерживаются архитектурой X.org/DRI, поэтому мы вынуждены использовать закрытые драйверы, выпускаемые изготовителем.

Драйвер от NVIDIA (*http://www.nvidia.com/*), кажется, пока не использует DRI, но в общих чертах дизайн драйвера весьма близок к этой инфраструктуре. Драйвер поставляется в виде готового к запуску двоичного файла инсталлятора, который компилирует и устанавливает модуль ядра (что соответствует драйверу DRM в DRI), затем устанавливает драйвер XAA двухмерной графики в X.org и заменяет системную библиотеку *libGL* поставляемой NVIDIA. Обратите внимание: модуль ядра поставляется в исходных текстах, но другие два компонента – только в виде двоичных файлов. За дополнительной информацией об установке драйвера NVIDIA обращайтесь на веб-сайт компании.

Компания ATI *http://www.ati.com/* также предоставляет драйвер для современных трехмерных графических ускорителей своего производства. Но, в отличие от NVIDIA, этот драйвер фактически использует инфраструктуру DRI. Во всем остальном эти драйверы напоминают друг друга: модуль ядра поставляется с исходными текстами, драйвер X.org и драйвер DRI – только в виде двоичных файлов, а также замена библиотеки *libGL*.

Настройка OpenGL в X.org

При наличии всех компонентов, составляющих поддержку OpenGL, и соответствующих драйверов можно настроить систему для ее использования.

Диагностика. Программа *glxinfo* представляет собой ценный инструмент, используемый в процессе настройки OpenGL в X.org, который позволяет получить информацию о поддержке OpenGL на текущем дисплее. Ниже приводится пример результатов, возвращаемых программой *glxinfo*:

```
$ glxinfo|less
name of display: :0.0
display: :0 screen: 0
direct rendering: Yes
server glx vendor string: SGI
server glx version string: 1.2
server glx extensions:
GLX_ARB_multisample, ...
client glx vendor string: SGI
client glx version string: 1.4
client glx extensions:
GLX_ARB_get_proc_address, ...
GLX extensions:
GLX_ARB_get_proc_address, ...
OpenGL vendor string: Tungsten Graphics, Inc.
OpenGL renderer string: Mesa DRI Radeon 20030328 AGP 1x x86/MMX/SSE2 TCL
OpenGL version string: 1.2 Mesa 6.1
OpenGL extensions:
...
```

Этот листинг показывает, что в настоящее время используется режим прямой визуализации с помощью драйвера DRI, основанного на Mesa, для графической видеокарты ATI Radeon. Если бы аппаратное ускорение не было настроено должным образом или что-то не работало бы, то на экране появилась бы строка *direct rendering: No* (режим прямой визуализации: Нет).

Внесение изменений в xorg.conf. Чтобы заставить работать инфраструктуру DRI, необходимо добавить пару строк[1] в файл с настройками *xorg.conf*, о котором рассказывалось выше в этой главе:

```
Section "Module"
...
# Загрузка модуля GLX
Load "glx"
# Загрузка модуля DRI
Load "dri"
EndSection
...
Section "DRI"
Mode 0666
EndSection
```

[1] Или проверить их наличие после автоматической инсталляции Linux, как обычно и бывает для всех новых дистрибутивов. – *Примеч. науч. ред.*

Директивы `Load` выполняют загрузку модулей, необходимых для работы OpenGL в X-сервере, а директива `Mode` в разделе DRI устанавливает биты прав доступа к специальному файлу символьного устройства, который используется приложениями для взаимодействия с драйвером DRM в ядре. Специальный файл устройства – это */dev/dri/card*`N`. Численное выражение прав доступа 0666 позволяет всем пользователям, обладающим доступом к дисплею X, использовать возможности OpenGL с аппаратным ускорением.

Mesa

Mesa – это библиотека трехмерной графики с прикладным интерфейсом, очень близко напоминающим OpenGL. Ядро библиотеки Mesa лицензируется в соответствии с авторским правом на X.org (MIT-подобная лицензия). Библиотека Mesa включается в состав X.org вместе с инфраструктурой DRI, но если у вас появится желание использовать OpenGL на платформе, не поддерживающей DRI, или попробовать свои силы в программировании для OpenGL, можно порекомендовать установить отдельную копию Mesa хотя бы для того, чтобы получить исходные тексты примеров программ.

Установка Mesa. Если по каким-либо причинам необходимо скомпилировать свою копию библиотеки Mesa, тогда просто загрузите самую свежую версию библиотеки MesaLib (вместе с примерами программ MesaDemos) с сайта *http://www.mesa3d.org/*, распакуйте и скомпилируйте. В текущей версии Mesa не используется инструмент GNU `autoconf`, вместо него поставляется `Makefile`, который содержит цели для сборки библиотеки под большое число комбинаций аппаратного окружения и операционных систем:

```
$ tar xfj MesaLib-6.2.1.tar.bz2
$ tar xfj MesaDemos-6.2.1.tar.bz2
$ cd Mesa-6.2.1
$ make             Выведет список поддерживаемых целей (платформ)
$ make linux-x86   Мы выбрали Linux/x86
```

По окончании компиляции попробуйте запустить некоторые из демонстрационных приложений, чтобы проверить правильность сборки.

```
$ cd lib
$ export LD_LIBRARY_PATH=$PWD:$LD_LIBRARY_PATH
$ cd ../progs/demos
$ ./gears
```

Если демонстрационное приложение работает, можно выполнить установку библиотеки:

```
$ cd ../../
$ cp -r include/GL /usr/local/mesa/include Установка заголовочных файлов
$ cp -d lib/* /usr/local/mesa/lib          Установка библиотек
```

Спрятав заголовочные файлы и библиотеки в каталог */usr/local/mesa*, впоследствии можно будет легко и просто переключаться между поставляемой в составе системы реализацией OpenGL и Mesa за счет изменения переменной окружения `LD_LIBRARY_PATH`, включая в нее или исключая путь к каталогу */usr/local/mesa/lib*.

17

Запуск и останов системы

Загрузка системы

Существует несколько способов загрузки Linux на машине. Чаще всего загрузка происходит с жесткого диска или загрузочной дискеты. Во многих случаях процедура установки настраивает один или оба эти способа. В любом случае важно знать, как самостоятельно настроить процесс загрузки.

Использование загрузочной дискеты

Традиционно загрузочная дискета Linux просто содержит образ ядра, который загружается в память, когда вы вставляете дискету и запускаете машину.[1]

Многие дистрибутивы Linux создают загрузочную дискету во время установки системы. С помощью загрузочной дискеты легко запустить систему, если вы не хотите мучиться с загрузкой с жесткого диска. (Например, начальный загрузчик Windows NT/2000 не так просто настроить для загрузки Linux. Об этом мы поговорим в следующем разделе.) Как только ядро загружено с дискеты, можно использовать привод гибких дисков для других целей.

Мы поместили в книгу некоторые технические сведения, чтобы объяснить процедуру начальной загрузки, но можете поверить, что в большинстве случаев достаточно вставить в машину дискету, и система загрузится. И все же чтение последующих разделов поможет вам лучше разобраться в своей машине.

Образ ядра обычно сжат с использованием того же алгоритма, что используется программами сжатия *gzip* или *bzip2* (подробнее об этом см. в разделе «Сборка ядра» главы 18). Сжатие позволяет уместить ядро размером более 1 Мбайт в несколько сотен килобайт дискового пространства. Часть программного кода ядра не сжата – в ней содержатся программы, необходимые для разархивирования ядра из дискового образа и загрузки в память. Поэтому ядро фактически загружает себя, разархивируясь в память.

---

[1] Возможен другой вариант: загрузочная дискета Linux содержит загрузочную запись GRUB, которая заставляет систему загрузить ядро с жесткого диска. Мы обсудим это в следующем разделе, когда будем говорить о загрузчике GRUB.

В образе ядра содержится ряд параметров. Среди них имя устройства, используемого в качестве корневой файловой системы после загрузки ядра. Другим параметром является текстовый режим, используемый системной консолью. Все эти параметры можно изменить с помощью команды `rdev`, которую мы будем обсуждать ниже в этом разделе.

После запуска ядро пытается монтировать файловую систему на корневое устройство, жестко прошитое в самом образе ядра. Она будет служить корневой файловой системой, то есть файловой системой на /. В разделе «Управление файловыми системами» главы 10 о файловых системах рассказано более подробно, а сейчас вам достаточно знать, что в образе ядра должно содержаться имя устройства вашей корневой файловой системы. Если ядро не может смонтировать на это устройство файловую систему, оно сдается и выводит сообщение ядра «panic». (По сути, *kernel panic* – это неустранимая ошибка, о которой сообщает само ядро. Паника происходит, когда ядро находится в полном замешательстве и не может продолжать работу. Например, если в ядре есть программная ошибка, то паника может произойти при попытке доступа к несуществующей памяти. О панике ядра мы еще поговорим в разделе «Действия в аварийных ситуациях» главы 27.)

Корневое устройство, записанное в образе ядра, – это устройство вашей корневой файловой системы на жестком диске. Это значит, что после загрузки ядро монтирует раздел жесткого диска в качестве корневой файловой системы, и все управление переходит к жесткому диску. Будучи загруженным в память, ядро остается в ней – доступа к загрузочной дискете больше не потребуется (до тех пор, пока вам не потребуется перезагрузить систему).

Если у вас есть образ ядра не слишком большого размера, вы можете создать собственную загрузочную дискету. Во многих системах Linux само ядро хранится в файле */boot/vmlinuz*.[1] Однако это не общепринятое соглашение; прочие системы Linux хранят ядро в */vmlinuz* или */vmlinux* или в таком файле, как */Image*. (Если у вас несколько образов ядра, можно с помощью GRUB выбрать нужное для загрузки. GRUB будет описан в следующем разделе.) Учтите, что вновь установленная система Linux может не иметь на жестком диске образа ядра, если программа установки создала загрузочную дискету. В любом случае вы можете скомпилировать собственное ядро. Обычно это правильное решение: вы можете «настроить» ядро, чтобы оно включало только те драйверы, которые необходимы для ваших устройств. Подробности в разделе «Сборка ядра» главы 18.

Отлично. Допустим, у вас есть образ ядра в файле */boot/vmlinuz*. Для создания загрузочной дискеты нужно первым делом с помощью команды *rdev* установить корневое устройство на вашу корневую систему Linux. (Если вы сами компилировали ядро, то оно уже имеет правильное значение, хотя не повредит проверить его с помощью *rdev*.) О том, как создать корневое устройство, мы говорили в разделе «Редактирование /etc/fstab» главы 2.

---

[1] Откуда взялось такое странное название? Во многих системах UNIX ядро хранится в файле */vmunix*, где *vm* происходит от «virtual memory» (виртуальная память). Естественно, Linux должна быть оригинальной и называет образы своего ядра *vmlinux*, помещая их в каталог */boot*, чтобы не оставлять в корневом каталоге. Имя *vmlinuz* было принято для отличия сжатых образов от несжатых. На практике имя и местонахождение ядра не играют никакой роли, если у вас есть загрузочная дискета с ядром или GRUB знает, где найти его образ.

Зарегистрировавшись как *root*, введите команду *rdev -h*, чтобы просмотреть сообщение об использовании команды. Как вы увидите, поддерживается много параметров, позволяющих задать корневое устройство (наша текущая задача), устройство подкачки, размер электронного диска и т. д. Эти параметры вас сейчас в основном не должны волновать.

Если мы выполним команду *rdev /boot/vmlinuz*, то будет выведено корневое устройство, зашитое в ядре, находящемся в */boot/vmlinuz*:

```
courgette:/# rdev /boot/vmlinuz
Root device /dev/hda1
```

Если это неверно и корневая файловая система Linux фактически лежит на */dev/hda3*, следует выполнить команду:

```
courgette:/# rdev /boot/vmlinuz /dev/hda3
courgette:/#
```

У *rdev* характер сильный и немногословный: когда устанавливается корневое устройство, не выводится никаких сообщений, поэтому снова выполните команду *rdev /boot/vmlinuz*, чтобы проверить правильность установки.

Теперь вы можете создать загрузочную дискету. Лучше всего использовать новую отформатированную дискету. Можно отформатировать дискету под Windows или с помощью `fdformat` в Linux; при этом будет записана информация о секторах и дорожках, чтобы система могла автоматически определить размер дискеты. (Дополнительную информацию о форматировании дискет можно найти в разделе «Управление файловыми системами» главы 10.)

Чтобы создать загрузочную дискету, воспользуйтесь командой *dd*, которая скопирует на дискету образ ядра:

```
courgette:/# dd if=/boot/vmlinuz of=/dev/fd0 bs=8192
```

Если вас заинтересовала команда *dd*, можете почитать о ней на страницах справочного руководства. Если сказать кратко, то она копирует входной файл (параметр `if`) с именем */boot/vmlinuz* в выходной файл (параметр `of`) с именем */dev/fd0* (первый привод гибких дисков), используя блок размером (`bs`) 8192 байт. Конечно, можно прибегнуть к плебейской `cp`, но системные администраторы UNIX любят использовать таинственные команды для выполнения сравнительно простых задач. Это то, что отличает нас от простых смертных.

Теперь ваша загрузочная дискета должна быть готова к работе. Вы можете остановить систему (подробнее об этом рассказывается в разделе «Останов системы» далее в этой главе) и загрузиться с дискеты, и если все пойдет хорошо, ваша Linux-система должна загрузиться как обычно. Неплохо сделать про запас еще одну загрузочную дискету, а в разделе «Действия в аварийных ситуациях» главы 27 мы опишем способы использования загрузочных дискет при восстановлении после катастрофы.

Использование GRUB

GRUB является менеджером загрузки общего назначения, способным загружать любые операционные системы, установленные на вашей машине, в том числе Linux. Есть десятки способов конфигурирования GRUB. Мы хотим изложить два

наиболее часто используемых метода: установку GRUB в главную загрузочную запись жесткого диска и установку GRUB как вторичного загрузчика только для Linux.

Чаще всего GRUB используется для загрузки Linux с жесткого диска. (Под загрузкой с жесткого диска мы подразумеваем, что само ядро хранится на жестком диске и загрузочная дискета не нужна. Но не забывайте, что, даже если вы пользуетесь загрузочной дискетой, как только ядро загружено в память, контроль передается жесткому диску.) Если загрузчик GRUB установлен в главную загрузочную запись вашего диска (MBR), то это первая программа, которая выполняется при загрузке жесткого диска. Затем GRUB может загрузить другие операционные системы, например Linux или Windows, и позволить вам делать выбор во время загрузки.

Следует отметить, что GRUB – не единственный менеджер загрузки, доступный для запуска Linux. Есть альтернативы, такие как более старый LILO (Linux Loader – загрузчик Linux), который работает так же хорошо. Однако из-за того, что большинство дистрибутивов используют GRUB, именно этот загрузчик мы здесь и рассматриваем.

Однако в Windows NT и последующих версиях Windows есть собственные загрузчики, располагающиеся в MBR. Если вы пользуетесь одной из этих систем, можно установить GRUB в качестве «вторичного» начального загрузчика только для Linux. В этом случае GRUB устанавливается в загрузочную запись только раздела Linux, а начальный загрузчик Windows NT/2000/XP берет на себя задачу запустить оттуда GRUB, когда вы хотите загрузить Linux.

Однако, как мы увидим, начальные загрузчики Windows NT/2000 не очень расположены к сотрудничеству, когда дело касается загрузки GRUB. Это слабое архитектурное решение, и если вам абсолютно необходимо использовать один из этих начальных загрузчиков, проще загружать Linux с дискеты. Либо, если вы действительно готовы больше не расставаться с Linux, используйте GRUB для загрузки Windows NT/2000/XP и вообще забудьте о начальных загрузчиках Windows. Обычно этот путь довольно безболезненный, и именно его мы рекомендуем пройти. К тому же большинство дистрибутивов устанавливают загрузчик автоматически, если на компьютере, куда устанавливается Linux, уже присутствует Windows.

Использовать GRUB с Windows 95/98/ME/2000/XP совсем просто. Нужно лишь настроить GRUB для загрузки Windows 95/98/ME/2000/XP (описание вы найдете в следующем разделе). Однако, если вы установите Windows 95/98/ME/2000/XP после установки GRUB, вам понадобится переустановить GRUB, поскольку процедура установки Windows 95/98/ME/2000/XP перепишет MBR первичного жесткого диска. Позаботьтесь о том, чтобы иметь на руках загрузочную дискету Linux для загрузки Linux и повторного запуска GRUB.

Прежде чем продолжить, следует отметить, что ряд дистрибутивов Linux может устанавливать и настраивать GRUB во время первоначальной установки Linux. Однако часто лучше всего позволить программе установки дистрибутива выполнить первичную настройку GRUB, а затем самостоятельно донастроить GRUB под свои нужды.

Файл /etc/grub.conf

Первым шагом в настройке GRUB является установка файла с настройками GRUB, который обычно находится в */etc/grub.conf*. Файл */etc/grub.conf* содержит ссылки на другие файлы, с которыми мы познакомимся позже. Чаще всего файл */etc/grub.conf* имеет небольшой размер.

Мы рассмотрим образец файла *grub.conf*. Вы можете использовать его в качестве основы для собственного *grub.conf* и отредактировать для использования в своей системе.

Следует заметить, что загрузчик GRUB обладает значительной гибкостью и фактически предоставляет в распоряжение пользователя командную оболочку, в которой можно вводить команды GRUB в процессе загрузки. Однако большинство пользователей считают это утомительным занятием, сопряженным с появлением ошибок, и именно по этой причине здесь будет описываться иной способ использования загрузчика, при котором в ваше распоряжение будет предоставлено удобное меню, с помощью которого можно будет, например, выбирать, какое ядро или даже какую операционную систему следует загрузить. Файл *grub.conf*, содержащий все необходимые настройки, может быть довольно коротким, например:

```
root (hd0,0)
install --stage2=/boot/grub/stage2 /grub/stage1 (hd0) /grub/stage2 0x8000
     (hd0,0)/grub/menu.lst
quit
```

Первая строка определяет устройство, с которого будет производиться загрузка. В данном случае это первый раздел первого жесткого диска, `hd` – это жесткий диск (hard drive). Первый нуль означает первый жесткий диск, второй нуль – первый раздел заданного диска (GRUB всегда начинает счет с нуля!). В среде пользователей Linux принято резервировать небольшой раздел диска (часто монтируется в каталог */boot*), где должны храниться образ ядра и файлы, имеющие отношение к загрузке. Поскольку ранее существовали некоторые ограничения на разделы дисков, откуда загрузчик мог бы загрузить образ ядра, стало общепринятым делать этот раздел самым первым. Хотя с появлением современных аппаратных средств, загрузчиков и базовых систем ввода-вывода (BIOS) эти ограничения давно уже сняты, тем не менее данная традиция до сих пор живет, и не будет никакого вреда, если она сохранится и впредь.

Еще несколько примеров: обозначение `(fd0)` означало бы необходимость загрузки с первого гибкого диска, а `(hd3,4)` будет означать пятый раздел на четвертом жестком диске (одного удачливого пользователя Linux, кто смог позволить себе такое количество дисков). Кроме того, существует еще одно специальное обозначение – `(nd)`, которое используется для загрузки образа ядра из сети, но описание этой возможности уже выходит за рамки данной книги.

Вторая строка значительно сложнее. Мы зашли бы слишком далеко, если бы взялись за описание всех подробностей процесса загрузки, но мы можем хотя бы сказать, что GRUB реализует двухступенчатый процесс загрузки и данная строка описывает, где находятся команды этих двух стадий, которые должны быть загружены в память компьютера, и какие действия следует выполнять затем. «Что выполнять затем» – это самая интересная для нас часть. В примере файла с настройками эта часть находится в самом конце строки – `(hd0,0)/grub/menu.lst`.

Где должен находиться этот файл, чтобы загрузчик GRUB смог отыскать его? Если попробовать дать команду:

```
pooh:~ # ls /grub/menu.lst
/bin/ls: /grub/menu.lst: No such file or directory
```

файл не будет найден. Но вспомните, GRUB использует круглые скобки, чтобы указать требуемое устройство, в данном случае раздел жесткого диска – первый раздел первого диска. Разумеется, это мог бы быть любой раздел любого диска, но, если вы помните, большинство пользователей создают небольшой раздел специально для хранения образа ядра и некоторого количества файлов загрузчика, который обычно монтируется в каталог */boot*, поэтому можно попробовать дать такую команду:

```
pooh:~ # ls /boot/grub/menu.lst
/boot/grub/menu.lst
```

Вот он где! Конечно же, если вы, в отличие от нас, будете устанавливать GRUB на пустом месте, в системе не будет предустановленного файла с настройками, и тогда вы не сможете получить результат, подобный этому. Однако будьте уверены, именно в этом месте GRUB станет искать файл меню.

На что похож такой файл меню? Ниже приводится пример файла меню, который загружает две различных конфигурации системы Linux и MS-Windows, позволяя выбирать необходимое из удобного меню:

```
default 0
timeout 10

title Linux
    kernel (hd0,5)/boot/vmlinuz root=/dev/hda6 vga=0x314

title Linux Failsafe
    kernel (hd0,5)/boot/vmlinuz root=/dev/hda6 ide=nodma apm=off acpi=off vga=normal
noresume barrier=off nosmp noapic maxcpus=0 3

title Windows
    root (hd0,0)
    chainloader +1
```

Первые две строки означают, что после того, как на экране появится меню, у пользователя будет 10 секунд (`timeout`), чтобы сделать свой выбор. В противном случае будет загружена операционная система, соответствующая первой записи в списке, который следует ниже.

Вслед за этими начальными строками располагаются три раздела, каждый из которых начинается со слова `title` (заголовок). Вслед за каждым словом `title` располагаются строки, содержащие текст, который будет отображаться в меню загрузки. Для обеих конфигураций Linux строка, начинающаяся со слова `kernel` (ядро), описывает, откуда и какой образ ядра будет загружен. Все, что следует далее в этих строках, передается непосредственно ядру в качестве дополнительных параметров загрузки, включая корневое устройство и режим терминала (`vga=0x314`). В так называемой безаварийной (failsafe) конфигурации определяется множество дополнительных параметров, которые отключают практически все функциональные возможности ядра, способные привести к отказам. В такой конфигурации система не будет отличаться высоким быстродействием и полнотой функцио-

нальных возможностей, но в аварийных ситуациях наличие безаварийного ядра позволит загрузить систему и выполнить восстановительные работы.

Если бы эти команды приходилось вводить вручную, в командной оболочке GRUB по окончании ввода потребовалось бы дать команду `boot`. Команда `kernel` загружает ядро в память, но не запускает загрузку системы, а вот команда `boot` инициирует процесс загрузки системы. Однако, когда используется система меню GRUB, как в данном примере, команда `boot` запускается неявно, как если бы она присутствовала в конце каждого раздела.

Загрузка операционной системы производится несколько иначе. Загрузчик GRUB не способен напрямую загружать другие операционные системы, за исключением Linux, операционных систем семейства BSD и некоторых других. Для других систем, таких как Windows, вызывается загрузчик, устанавливаемый вместе с этими системами. Это называется *загрузкой по цепочке* (*chain-loading*), и потому совершенно не удивительно, что команда GRUB, которая делает это, называется `chainloader`. Параметр `+1` означает, что начальный загрузчик другой операционной системы находится в первом секторе раздела, который ранее был определен командой *root*.

По окончании настройки GRUB его нужно установить. Проще всего сделать это с помощью команды *grub-install*, которой передается имя каталога, где находятся файлы со стадиями и образы ядра, а также имя устройства, куда следует установить начальный загрузчик. Типичная команда выглядит следующим образом:

```
pooh:~ # grub-install --root-directory=/boot /dev/hda
```

Эта команда устанавливает начальный загрузчик на первый жесткий диск IDE (`/dev/hda`), где обычно BIOS компьютера пытается отыскать информацию, необходимую для выполнения загрузки.

Все, что было сказано выше, это лишь краткое введение в GRUB. Загрузчик GRUB может намного больше, например загружать образы ядра из сети или по последовательной линии, отображать меню в графическом режиме и многое другое. Если предполагается делать с помощью GRUB что-то более серьезное (так как любой загрузчик является ключом к использованию системы вообще, любые изменения в файлах с настройками, помимо таких тривиальных, как изменение строк пунктов меню, выбор пункта по умолчанию или изменение времени ожидания, в этом контексте можно считать серьезными), мы настоятельно рекомендуем предварительно ознакомиться с прекрасной документацией GRUB, которая поставляется в виде страниц Info и может быть вызвана для прочтения в окружении рабочего стола KDE вводом строки `info:grub` в панели адреса броузера Konqueror.

Определение дополнительных параметров во время загрузки

Скорее всего, сразу после установки Linux вы будете загружать систему с гибкого диска или с CD-ROM, где, весьма вероятно, загрузка будет выполняться загрузчиком GRUB (или другим), который предоставит меню загрузки. Выбрав требуемый пункт и нажав клавишу e, вы попадете в командную строку загрузчика. В этой строке можно будет ввести некоторые дополнительные параметры загрузки, например:

```
hd=cylinders,heads,sectors
```

которая описывает геометрию жесткого диска. Каждый раз при загрузке Linux может потребоваться вводить эти значения, чтобы определить корректные параметры аппаратного обеспечения компьютера. Если GRUB используется для загрузки Linux с жесткого диска, эти параметры можно указать в строке `kernel` файла с настройками GRUB и избавить себя от необходимости вводить их вручную каждый раз. Для этого нужно лишь добавить примерно такую строку в раздел с описанием загрузки Linux:

```
append = "hd=683,16,38"
```

Эта строка соответствует команде `hd=683,16,38`, введенной в командной строке GRUB. Если необходимо добавить несколько параметров загрузки, сделать это можно в одной строке `append`, например:

```
append = "hd=683,16,38 hd=64,32,202"
```

В этой ситуации определяется геометрия первого и второго жесткого диска, соответственно. После того как ввод команд будет закончен, можно нажать клавишу Esc, чтобы вернуться обратно в меню и начать загрузку системы.

Обратите внимание: ввод дополнительных параметров может потребоваться только в том случае, если ядро оказывается не в состоянии определить параметры оборудования на этапе загрузки, что весьма маловероятно, разве что если в эксплуатации находится довольно старое или очень редко встречающееся оборудование. О необходимости таких манипуляций вы должны уже знать по опыту установки Linux. Вообще, определять параметры ядра в файле меню GRUB может потребоваться, только если вам пришлось вводить их вручную сразу после установки Linux.

Существует множество других параметров загрузки. Большинство из них связаны с обнаружением аппаратных средств, о чем уже говорилось в главе 2. Тем не менее следующие параметры могут оказаться необходимыми для вас:

`single`

Загрузка системы в однопользовательском режиме – пропускает все настройки системы и запускает командную оболочку с привилегиями пользователя *root* в консоли. О том, как используется этот режим, мы расскажем в разделе «Действия в аварийных ситуациях» главы 27.

`root=`*`partition`*

Монтирует раздел *`partition`* как корневую файловую систему Linux.

`ro`

Монтирует корневую файловую систему в режиме только для чтения. Обычно это делается перед запуском утилиты `fsck`, о чем подробнее будет рассказываться в разделе «Действия в аварийных ситуациях» главы 27.

`ramdisk=`*`size`*

Указывает размер в байтах для устройства электронного диска. Большинству пользователей едва ли придется беспокоиться по поводу использования электронного диска, поскольку обычно это бывает необходимо только для нужд установки.

`vga=mode`

Определяет режим VGA работы монитора. Корректными режимами являются `normal`, `extended`, `ask` и числовые значения.

`mem=size`

Сообщает ядру объем имеющегося в наличии ОЗУ. Если в компьютере установлено 64 Мбайт памяти или меньше, ядро сможет получить эту информацию непосредственно от BIOS, но если используется достаточно старое ядро и объем памяти больше 64 Мбайт, может потребоваться сообщить ядру этот факт, в противном случае оно будет использовать только первые 64 Мбайт памяти. Например, допустим, что в компьютере имеется 128 Мбайт ОЗУ, тогда необходимо определить параметр `mem=128m`. К счастью, при использовании современных ядер необходимость в этом отпала.

Любые из этих параметров можно ввести вручную из командной строки GRUB или указать в строке `kernel` файла с настройками GRUB.

Удаление GRUB

Если загрузчик GRUB был установлен в главную загрузочную запись (MBR), самый простой способ избавиться от него – это воспользоваться программой *FDISK*, входящей в состав операционных систем Windows 95/98/ME:

```
FDISK /MBR
```

Эта команда запишет загрузочную запись Windows в MBR. В Windows NT/2000/XP эта процедура выполняется намного сложнее.

Запуск и инициализация системы

В этом разделе мы расскажем, что именно происходит при загрузке системы. Понимание этого процесса и участвующих в нем файлов важно для осуществления разного рода настроек системы.

Сообщения ядра при загрузке

На первом шаге загружается ядро. Как сказано в предыдущем разделе, загрузка может происходить с дискеты или жесткого диска. По мере загрузки в память ядро выводит различные сообщения на системную консоль, а также обычно записывает их в системные журналы. Зарегистрировавшись как *root*, вы всегда можете проверить файл */var/log/messages* (в котором содержатся также и сообщения ядра, выдаваемые во время работы). Команда *dmesg* выводит последние сообщения, содержащиеся в кольцевом буфере сообщений ядра. Сразу после загрузки вы, естественно, получите сообщения загрузки.

В следующих нескольких разделах мы разберем ряд наиболее интересных сообщений и разъясним их смысл. Все эти сообщения выводит само ядро по ходу инициализации каждого драйвера устройства. Точный текст сообщений зависит от того, какие драйверы включены в ядро и какие устройства есть на машине. Вполне возможно, что у вас сообщений будет больше, меньше или они будут другими. Мы остановим свое внимание на тех сообщениях, которые встречаются очень часто.

Строка:

```
Linux version 2.6.11.4-default (geeko@buildhost) (gcc version 3.3.5 2005011
7 (prerelease) (SUSE Linux)) #1 Thu Jun 2 14:23:14 UTC 2005
```

сообщает номер версии ядра, имя машины, время и компилятор, с помощью которого оно было собрано.

Затем ядро сообщает ряд сведений о BIOS, об объеме имеющейся памяти, о настройках управления питанием и т. п. Мы приведем лишь наиболее интересные (разумеется, в зависимости от имеющегося оборудования эти строки могут существенно отличаться):

```
...
127MB HIGHMEM available.
896MB LOWMEM available.
...
Kernel command line: root=/dev/hda6 vga=0x314 selinux=0 splash=silent resume=/de
v/hda5
...
Detected 599.481 MHz processor.
...
```

Обратите внимание на строку с параметрами ядра – с ее помощью можно лишний раз убедиться, что ядро фактически было загружено с нужными параметрами.

Затем ядро сообщает, какие параметры настройки консоли были выбраны и какой тип консоли обнаружен:

```
Console: colour dummy device 80x25
```

Обратите внимание, что это относится только к текстовому режиму, используемому ядром, а не к возможностям видеокарты. Эта информация вообще никакого отношения не имеет к X Window System – ядро вообще не интересуется подобными сведениями.

Следующее выведенное сообщение – расчет «BogoMips» процессора:

```
Calibrating delay loop... 1187.84 BogoMIPS (lpj=593920)
```

Это совершенно фальшивое измерение скорости процессора (отсюда и название), используемое для получения оптимальной производительности в циклах задержки некоторыми драйверами устройств.

Затем ядро собирает сведения о шине PCI и проверяет наличие в системе карт PCI:

```
PCI: PCI BIOS revision 2.10 entry at 0xfd8d6, last bus=8
PCI: Using configuration type 1
...
PCI: Probing PCI hardware (bus 00)
PCI: Ignoring BAR0-3 of IDE controller 0000:00:1f.1
PCI: Transparent bridge - 0000:00:1e.0
...
```

Затем инициализируются сетевые возможности, порт мыши и драйвер последовательных портов. Строка

```
ttyS00 at 0x03f8 (irq = 4) is a NS16550A
```

означает, что первое последовательное устройство (*/dev/tty00*, или COM1) было обнаружено по адресу 0x03f8, IRQ 4 и использует функции UART 16550A. Затем идет поиск некоторых других аппаратных устройств, например датчика истинного времени и привода гибких дисков:

```
Real Time Clock Driver v1.12
...
Floppy drive(s): fd0 is 1.44M
FDC 0 is a National Semiconductor PC87306
loop: loaded (max 8 devices)
...
```

Строка

```
Adding 1029632k swap on /dev/hda5. Priority:42 extents:1
```

сообщает объем пространства для свопинга, обнаруженного ядром. В число других задач, осуществляемых при типичной загрузке, входят обнаружение и настройка параллельного порта (`lp1`), обнаружение и настройка сетевой карты и, наконец, настройка подсистемы USB.

Опять же, в зависимости от версии ядра, параметров настройки и аппаратного окружения, вы можете столкнуться с другими сообщениями (в конце концов, то, что было показано здесь, – это всего лишь короткая выборка).

Файлы init, inittab и rc

После инициализации драйверов устройств ядро выполняет программу *init*, находящуюся в */etc*, */bin* или */sbin* (в большинстве систем */sbin/init*). Программа *init* является многоцелевой; она порождает новые процессы и перезапускает некоторые программы, когда они заканчивают работу. Например, для каждой виртуальной консоли *init* запускает процесс *getty*. После регистрации процесс *getty* заменяется другим, а после выхода из системы *init* запускает новый процесс *getty*, позволяющий опять зарегистрироваться.

Программа *init* отвечает также за запуск ряда программ и сценариев при загрузке системы. Вся работа *init* управляется файлом */etc/inittab*. Чтобы понять его содержимое, нужно сначала разобраться с концепцией *уровней исполнения* (*runlevels*).

Уровень исполнения – это число или буква, обозначающая текущее состояние системы по отношению к *init*. Например, когда уровень исполнения изменяется на 3, будут выполнены все строки в */etc/inittab*, содержащие цифру 3 в колонке уровня исполнения. Уровни исполнения являются удобным способом группировки строк в */etc/inittab*. Например, вам бы хотелось, чтобы на уровне исполнения 1 выполнялся только самый минимум сценариев конфигурации, на уровне 2 выполнялось все то же, что на уровне 1, плюс настройка сети, на уровне 3 – все, что на уровнях 1 и 2, плюс удаленный вход по коммутируемым линиям, и т. д. Современные версии дистрибутивов Red Hat и SUSE настроены так, что на уровне исполнения 5 автоматически стартует графический интерфейс X Window System. Дистрибутив Debian делает это на уровнях исполнения со 2-го по 5-й при условии, что был установлен менеджер дисплея X – `xdm`.

Чаще всего уровни исполнения не должны вас беспокоить. При загрузке система выходит на уровень исполнения по умолчанию (устанавливаемый в */etc/inittab*,

как мы скоро покажем). В большинстве систем это уровень 3 или 5. После обсуждения обычной загрузки мы покажем, как выйти на другой уровень исполнения, который иногда требуется, – уровень исполнения 1, или однопользовательский режим. Пользователи Debian могут заняться исследованием пакета *file-rc*, который позволит настроить уровни исполнения в единственном файле.

Посмотрим на пример файла */etc/inittab*:

```
# Выбор уровня исполнения по умолчанию
id:3:initdefault:

# Первый сценарий, который будет запущен
si::bootwait:/etc/init.d/boot

# Запуск /etc/init.d/rc с номером уровня исполнения в качестве аргумента
l0:0:wait:/etc/init.d/rc 0
l1:1:wait:/etc/init.d/rc 1
l2:2:wait:/etc/init.d/rc 2
l3:3:wait:/etc/init.d/rc 3
l4:4:wait:/etc/init.d/rc 4
l5:5:wait:/etc/init.d/rc 5
l6:6:wait:/etc/init.d/rc 6

# Выполняется при нажатии комбинации ctrl-alt-delete
ca::ctrlaltdel:/sbin/shutdown -t3 -rf now

# Запуск agetty на виртуальных консолях с 1 по 6
1:2345:respawn:/sbin/mingetty --noclear tty1
2:2345:respawn:/sbin/mingetty tty2
3:2345:respawn:/sbin/mingetty tty3
4:2345:respawn:/sbin/mingetty tty4
5:2345:respawn:/sbin/mingetty tty5
6:2345:respawn:/sbin/mingetty tty6
```

Поля разделяются двоеточием. Легче всего узнать последнее поле: это команда, которую *init* выполняет для этой строки. Первое поле является произвольным идентификатором (не важно, каким, лишь бы он был уникален в этом файле), а второе поле определяет, на каком уровне исполнения должна быть запущена команда. Третье поле сообщает *init*, как обрабатывать строку; например, выполнить команду один раз или перезапускать после каждого завершения.

Точное содержание файла */etc/inittab* зависит от вашей системы и установленного вами дистрибутива Linux.

В нашем примере мы видим сначала, что по умолчанию выбран уровень исполнения 3. Поле `action` этой строки содержит `initdefault`, что влечет установление данного уровня исполнения в качестве значения по умолчанию. Это уровень исполнения, обычно действующий при загрузке системы. Вы можете переопределить значение по умолчанию, запустив *init* вручную (что можно сделать при отладке настроек) и передав желаемый уровень выполнения в качестве аргумента. Например, следующая команда завершает все текущие процессы и стартует с уровня выполнения 5 (прежде чем это сделать, предупредите всех пользователей о необходимости выйти из системы!):

```
tigger# init 5
```

GRUB тоже может загружать систему в однопользовательском режиме (обычно это уровень исполнения 1), подробнее об этом рассказывается в разделе «Определение дополнительных параметров во время загрузки» выше в этой главе.

Следующая строка сообщает *init* о необходимости выполнить при загрузке системы сценарий */etc/init.d/boot*. (В поле `action` стоит `si` [sysinit], что указывает на необходимость выполнения этой строки, когда *init* впервые запускается при загрузке системы.) В других дистрибутивах этот файл может находиться в другом месте, однако стандарт, описывающий структуру файловой системы Linux (Linux Filesystem Hierarchy Standard, FHS), диктует имя и местоположение этого файла как */etc/init.d/boot*. Этот файл является просто сценарием командной оболочки, содержащим команды для осуществления базовой инициализации системы. Например, включается свопинг, проверяются и монтируются файловые системы, и системные часы синхронизируются с таймером CMOS. Многие из команд, содержащихся в этом файле, уже обсуждались в разделах «Управление файловыми системами» и «Управление пространством свопинга» главы 10.

Далее мы видим, что система выполняет сценарий */etc/init.d/rc*, когда входит на любой из уровней исполнения от 0 до 6 с соответствующим уровнем исполнения в качестве аргумента. *rc* является типовым начальным сценарием, запускающим другие сценарии в соответствии с уровнем исполнения. Поле `action` содержит `wait`, что сообщает *init* о необходимости выполнить данную команду и дождаться окончания ее выполнения, прежде чем идти дальше.

Файлы rc

В Linux команды начального запуска сохраняются в файлах, имена которых содержат «rc», согласно старинному соглашению UNIX. Эти команды делают все необходимое для получения полностью функционирующей системы, например запускают серверы и демоны, о которых говорилось в разделе «Процессы» главы 10. Благодаря этим командам система готова выполнять задачи регистрации, электронной почты, веб-сервера или другие, в зависимости от того, какое программное обеспечение вы установили и пожелали запустить. Как было разъяснено в предыдущем разделе, файлы запускаются из */etc/inittab*. Команды являются стандартными командами оболочки, и можно, просмотрев различные файлы *rc*, увидеть, что они делают.

В этом разделе мы опишем структуру файлов *rc*, чтобы вы могли понять, откуда все запускается, и запускать или останавливать серверы вручную в тех редких случаях, когда они не хотят делать то, что вы от них требуете. В качестве модели мы будем использовать SUSE, однако, получив представление о том, что нужно искать, вы сможете найти соответствующие файлы в любом дистрибутиве Linux. Дистрибутив SUSE достаточно точно следует требованиям стандарта FHS, таким образом, структура файловой системы в других дистрибутивах будет напоминать ту, о которой говорится здесь. Linux FHS представляет собой не связанную с каким-либо дистрибутивом инициативу, имеющую целью определить стандартные имена каталогов и файлов для важнейших системных файлов. Дистрибутив Debian может служить примером еще одного дистрибутива, следующего стандарту FHS. В Red Hat *rc*-сценарий верхнего уровня – это */etc/rc.d/rc*.

В предыдущем разделе вы могли видеть, как */etc/inittab* запускает сценарий в различных обстоятельствах с различными числами от 0 до 6 в качестве аргумента. Числа соответствуют уровням исполнения, каждый из которых заставляет *rc*-файлы запускать разные наборы сценариев. Поэтому нашим следующим шагом будет поиск сценариев, соответствующих каждому уровню исполнения.

В соответствии с требованиями FHS, сценарии каждого уровня исполнения хранятся в каталогах */etc/init.d/rcN.d*, где *N* – номер запускаемого уровня исполнения. Так, для уровня исполнения 3 будут использоваться сценарии из */etc/init.d/rc3.d*. Опять же, в других дистрибутивах, как правило, есть маленькие отличия.

Загляните в какой-нибудь из этих каталогов – вы увидите ряд файлов вида *Snnxxxx* или *Knnxxxx*, где *nn* – число от 00 до 99, а *xxxx* – имя какой-либо системной службы. */etc/rc.d/rc* сначала выполняет сценарии с именами, начинающимися на *K*, которые останавливают работающие службы. Затем выполняются сценарии с именами, начинающимися на *S*, которые запускают новые службы.

Числа *nn* в именах используются для определения порядка выполнения сценариев: первыми выполняются сценарии с меньшими номерами. Имя *xxxx* используется просто для того, чтобы помочь определить, какой из системных служб соответствует сценарий. Это соглашение по именованию может показаться странным, но оно позволяет легко добавлять и удалять сценарии из этих каталогов и помогает */etc/rc.d/rc* автоматически выполнять их в нужный момент. Для редактирования стартовых сценариев удобно использовать редактор уровней исполнения с графическим интерфейсом, такой как *KSysV* в KDE (будет обсуждаться далее в этой главе).

Например, сценарий для инициализации сетевого взаимодействия может называться *S10network*, а сценарий для останова демона, который ведет системный журнал, может называться *K70syslog*. Если эти сценарии расположены в соответствующих каталогах */etc/init.d/rcN.d*, они будут запущены */etc/rc.d/rc* в порядке увеличения номеров при запуске или останове системы. Если по умолчанию система загружается на уровне исполнения 3, загляните в */etc/rc.d/rc3.d*, чтобы увидеть, какие сценарии выполняются при нормальной загрузке системы.

Поскольку одни и те же службы запускаются или останавливаются на различных уровнях исполнения, дистрибутив SUSE использует символические ссылки вместо повторения одного и того же сценария в разных местах. Так, каждый *S*- или *K*-файл является символической ссылкой, указывающей на центральный каталог, в котором хранятся сценарии запуска и останова всех служб. Обычно это каталог */etc/init.d*. В SUSE и Debian в этом каталоге содержится сценарий с именем *skeleton*, который вы можете адаптировать для запуска и останова любых демонов, написанных вами.

Знать, где находится сценарий запуска или останова, полезно, если вы не хотите полностью перезагружаться или выходить на другой уровень исполнения, но вам нужно запустить или остановить некоторый сервис. Найдите в каталоге *init.d* сценарий с соответствующим именем и выполните его, передав параметр `start` или `stop`. Например, в SUSE, если необходимо запустить веб-сервер Apache, но система находится на уровне исполнения, в который не входит Apache, введите следующую строку:

```
tigger # /sbin/init.d/apache start
```

Многие дистрибутивы в самом конце запускают сценарий */etc/init.d/boot.local*. Вы можете редактировать файл *boot.local* так, чтобы выполнять любые специфические или в каком-то отношении неуместные во время загрузки команды, или если вы не знаете точно, в каком месте их следует выполнять. Парный ему сценарий, который запускается во время остановки системы, называется */etc/init.d/halt.local*.

Следующая строка с меткой `ca` выполняется при нажатии на консоли комбинации клавиш Ctrl+Alt+Del. Эта комбинация вызывает прерывание, которое обычно перезагружает систему. В Linux это прерывание перехватывается и посылается программе *init*, которая выполняет запись с `ctrlaltdel` в поле `action`. В данном случае здесь находится команда `/sbin/shutdown -t3 -rf now`, которая производит «безопасную» перезагрузку системы. (См. раздел «Останов системы» далее в этой главе.) Таким образом мы защищаем систему от случайной перезагрузки при нажатии Ctrl+Alt+Del.

И наконец, в файле *inittab* содержатся записи, которые запускают */sbin/mingetty* для первых шести виртуальных консолей. *mingetty* является одним из нескольких вариантов *getty*, имеющихся в Linux. Эти программы позволяют регистрироваться на терминалах; без них терминал был бы, в сущности, мертв и не реагировал на нажатие пользователем клавиши или кнопки мыши. Различные команды *getty* открывают терминальное устройство (например, виртуальную консоль или последовательную линию), устанавливают различные параметры драйвера терминала и запускают */bin/login* для начала сеанса регистрации на этом терминале. Поэтому для того, чтобы на данной виртуальной консоли можно было зарегистрироваться, на ней должна быть запущена программа *getty* или *mingetty*. В ряде систем Linux используется *getty* или *agetty*, имеющие несколько отличающийся синтаксис. За дополнительной информацией о *getty*, *agetty* и *mingetty* обращайтесь к соответствующим страницам справочного руководства.

mingetty принимает один аргумент – имя устройства. Порты для виртуальных консолей Linux называются */dev/tty1*, */dev/tty2* и т. д. *mingetty* предполагает, что имя устройства задается относительно каталога */dev*. Скорость обмена для виртуальных консолей обычно должна составлять 38 400. Это объясняет отсутствие данного значения при использовании *mingetty*, в отличие от программы *agetty*, которая требует явного указания значения скорости.

Обратите внимание, что для команды *mingetty* в поле `action` стоит параметр `respawn`. Это значит, что *init* необходимо перезапустить команду, указанную в записи, когда процесс *mingetty* завершится, что происходит каждый раз при завершении сеанса связи с пользователем.

Однопользовательский режим

Большую часть времени система работает в многопользовательском режиме, и пользователи могут регистрироваться в системе. Однако есть особое состояние, называемое *однопользовательским режимом*, когда UNIX работает, но приглашения к регистрации нет. В однопользовательском режиме вы, по существу, являетесь суперпользователем *root*. Вам может понадобиться войти в этот режим во время установки, если возникнут неполадки. Однопользовательский режим важен для некоторых рутинных административных задач, например проверки

поврежденных файловых систем. (Это нешуточное дело. Постарайтесь не повредить свою файловую систему. Например, всегда останавливайте систему командой останова перед отключением питания, как описано в следующем разделе.)

В однопользовательском режиме система почти бесполезна – используется малая часть конфигурации, файловые системы не монтируются и т. д. Это необходимо для восстановления после возникновения некоторых системных проблем (подробности в разделе «Действия в аварийных ситуациях» главы 27).

Обратите внимание: UNIX остается многозадачной системой даже в однопользовательском режиме. Одновременно могут выполняться несколько программ. Серверы могут выполняться в фоновом режиме, благодаря чему могут действовать некоторые специальные функции, такие как сетевое взаимодействие. Но если система поддерживает более одного терминала, можно использовать только консоль. Система X Window тоже не запустится.

Останов системы

К счастью, останов системы выполняется значительно проще, чем загрузка и запуск. Тем не менее это не просто нажатие кнопки reset. Linux, как и все системы UNIX, буферизует в памяти дисковые операции чтения и записи. Это значит, что запись на диск откладывается до того времени, когда она становится абсолютно необходимой, а чтение одного и того же блока диска осуществляется непосредственно из ОЗУ. Благодаря этому значительно увеличивается производительность, поскольку диски работают гораздо медленнее, чем процессор.

Проблема заключается в том, что если система внезапно отключается от питания или перезагружается, находящиеся в памяти буферы не будут записаны на диск, что может привести к потере или повреждению данных. Ядро выталкивает «грязные» буферы (то есть изменившиеся после чтения с диска) обратно на диск каждые 5 секунд, чтобы предотвратить серьезные повреждения в случае краха системы. Однако для полной надежности система должна до перезагрузки осуществить безопасный останов. Это не только обеспечит правильную синхронизацию дисковых буферов, но и позволит корректно завершиться всем выполняющимся процессам.

Команда *shutdown* является общей многоцелевой командой, используемой для останова или перезагрузки системы. Зарегистрировавшись как *root*, вы можете выполнить команду

```
/sbin/shutdown -r +10
```

чтобы заставить систему перезагрузиться через 10 минут. Параметр `-r` указывает на то, что система после останова должна быть перезагружена, а `+10` задает время ожидания до останова в минутах. Система выведет предупреждающее сообщение на все терминалы и начнет отсчет времени останова. Вы можете добавить собственное сообщение, включив его в командную строку:

```
/sbin/shutdown -r +10 "Перезагрузка для проверки нового ядра"
```

Можно задать и абсолютное время останова:

```
/sbin/shutdown -r 13:00
```

для перезагрузки в 1 час дня. Аналогично можно дать команду

```
/sbin/shutdown -r now
```

чтобы выполнить перезагрузку немедленно (после процесса безопасного останова).

Если вместо параметра *-r* использовать *-h*, то система будет просто остановлена без перезагрузки; после этого можно отключить питание, не опасаясь потери данных. Если не заданы ни *-h*, ни *-r*, система переходит в однопользовательский режим.

Как мы видели в разделе «Файлы init, inittab и rc», можно заставить *init* перехватывать комбинацию Ctrl+Alt+Del и в ответ на нее выполнять команду *shutdown*. Если вам привычнее перезагружать систему таким способом, будет далеко не лишним проверить наличие записи `ctrlaltdel` в файле */etc/inittab*. Обратите внимание: никогда нельзя перезагружать систему выключением питания или нажатием кнопки перезагрузки на передней панели машины. Если только система не зависла намертво (что бывает редко), всегда нужно использовать команду *shutdown*. Замечательное свойство многозадачной системы состоит в том, что если одна программа зависла, почти всегда можно переключиться в другое окно или виртуальную консоль, чтобы исправить положение.

Команда *shutdown* имеет ряд других параметров. Ключ *-c* прекратит текущий процесс останова. (Конечно, всегда можно прервать процесс с помощью *kill*, но *shutdown -c* может быть проще.) Ключ *-k* заставит вывести предупредительные сообщения, но не осуществит фактический останов системы. Подробности вы найдете на страницах справочного руководства *shutdown*(8).

Редактор уровней исполнения с графическим интерфейсом – KSysV

Если у вас создалось ощущение, что редактирование уровней исполнения простым копированием символических ссылок является слишком сложным и непонятным занятием, всегда можно прибегнуть к услугам редактора уровней исполнения с графическим интерфейсом. В состав практически любого дистрибутива входит какой-нибудь из подобных редакторов, но если было установлено окружение рабочего стола KDE, практически наверняка в вашем распоряжении имеется редактор KSysV.[1] Запустить программу можно либо из главного K-меню (System (Система)→Service Configuration (Редактор сценариев)→KSysV (KSysV)) при использовании типовых настроек KDE, либо командой `ksysv`.

При первом запуске KSysV задаст несколько вопросов о дистрибутиве, что поможет программе определить правильное местоположение ваших файлов настройки уровней исполнения. Затем на экране появится окно, показанное на рис. 17.1.

В левой части окна находится список доступных служб, в правой – два ряда списков служб, запускаемых на разных уровнях исполнения. Верхний ряд содержит

[1] Название произошло от того факта, что текущая система загрузки Linux, описывавшаяся в предыдущих разделах, изначально появилась в семействе операционных систем UNIX System V. Много лет назад в Linux использовалась иная система загрузки, в которой отсутствовало понятие уровней исполнения, но затем дистрибутивы Linux перешли на использование так называемой системы SysV.

списки, соответствующие уровням исполнения при запуске системы, нижний – при останове. Каждому уровню исполнения соответствует свой список в ряду (если уровень исполнения не был отключен флажком в строке состояния в нижней части окна, что очень удобно, когда требуется внести изменения только в определенные уровни исполнения).

Чтобы добавить службу на уровень исполнения, достаточно будет просто перетащить ее мышью из списка служб в левой части окна в требуемый список справа. Не забывайте добавлять службы как в список запуска, так и в список останова. Чтобы удалить службу с уровня исполнения, нужно захватить ее левой кнопкой мыши и перетащить к ярлыку мусорного ведра, что находится в левом нижнем углу окна программы. Кроме того, выбрав требуемую службу в любом из списков, можно выполнить ее настройку, а также остановить, запустить или перезапустить ее.

Когда все необходимые манипуляции со службами будут закончены, следует выбрать пункт меню File (Файл)→Save Configuration (Сохранить настройки), чтобы сохранить сделанные настройки. Однако операция сохранения может быть выполнена только при условии, что программа KSysV была запущена с привилегиями пользователя *root*, в противном случае файлы с настройками окажутся недоступными для записи. Однако, когда программа KSysV запускается в среде KDE, перед ее запуском будет предложено ввести пароль суперпользователя, и программа будет запущена с привилегиями пользователя *root*.

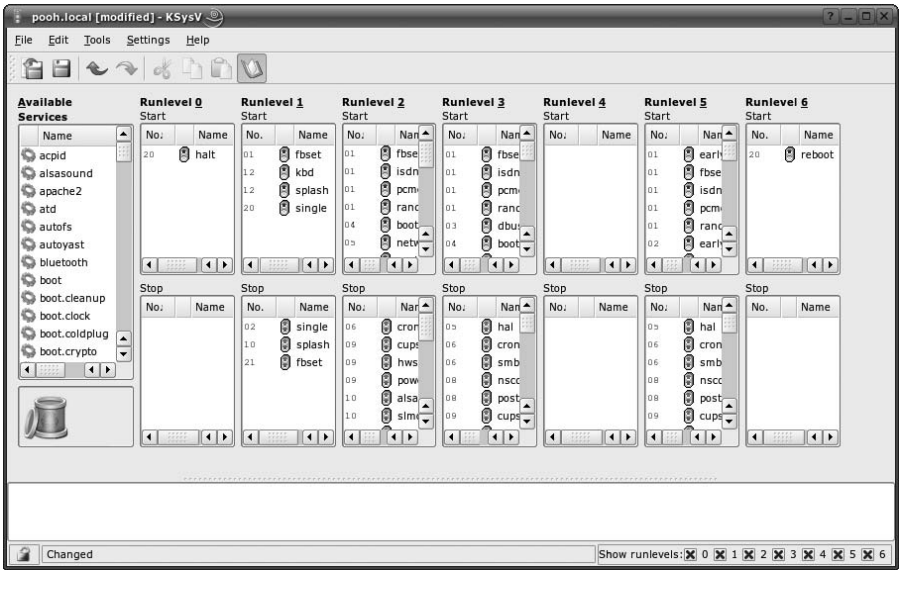


Рис. 17.1. *Главное окно программы KSysV*

18

Настройка и сборка ядра

Может показаться, что перекомпиляция ядра – занятие для хакеров, но на самом деле это умение необходимо любому системному администратору. Во-первых, заново компилировать ядро необходимо для того, чтобы избавиться от ненужных драйверов устройств. Благодаря этому сокращается объем памяти, используемой ядром, о чем говорилось в разделе «Управление пространством свопинга» главы 10. Ядро всегда присутствует в памяти, и занятую им память не могут использовать другие программы, когда им это требуется.

Во-вторых, большинство современных дистрибутивов строятся на основе модульных ядер. Это означает, что ядро, которое устанавливается по умолчанию, содержит в себе лишь минимально необходимые функциональные возможности, достаточные для того, чтобы запустить систему, а остальное содержится в виде модулей, реализующих все прочие функциональные возможности, которые только могут потребоваться. Более подробно о модулях ядра мы поговорим позже. Но даже при такой организации ядра производители дистрибутивов включают несколько версий ядра, чтобы обеспечить, например, поддержку однопроцессорных и многопроцессорных систем, поскольку такого рода возможности не могут быть реализованы в виде модулей. Программы установки, поставляемые в составе дистрибутивов, обычно достаточно интеллектуальны, чтобы выяснить, какое ядро нужно устанавливать, и автоматически делают оптимальный выбор.

Для чего же нужна возможность выбора разного рода модулей? Дело в том, что весь код ядра постоянно находится в памяти, то есть никакие участки памяти, принадлежащие ядру, не могут быть вытеснены в файл подкачки. Таким образом, если используется образ ядра, обладающего поддержкой фактически отсутствующих аппаратных средств, память, занимаемая ненужным кодом, будет недоступна пользовательским приложениям. Настройка ядра позволит уменьшить объем памяти, занимаемой ядром, и использовать ее для других нужд.

Кроме того, время от времени необходимо обновлять ядро, чтобы иметь более новую его версию. Как и в отношении другого программного обеспечения, когда становится известно об исправлении важных ошибок или добавлении новых функций в новую версию ядра, может потребоваться осуществить обновление, чтобы использовать их. Тем, кто активно разрабатывает код ядра, также требуется поддерживать ядро в текущем состоянии, если изменения в нем затрагива-

ют тот код, над которым они работают. Иногда необходимо обновить ядро, чтобы иметь возможность использовать новые версии компилятора или библиотек. Некоторые приложения (такие как эмулятор VMware) могут работать только с определенной версией ядра.

Выяснить версию работающего ядра можно с помощью команды *uname -a*. Она дает примерно такой вывод:

```
rutabaga$ uname -a
Linux pooh 2.6.11.4-21.7-default #1 Thu Jun 2 14:23:14 UTC 2005 i686 i686 i386 GNU/Linux
```

Откуда видно, что на машине используется ядро версии 2.6.11.4, скомпилированное 2 июня 2005 года. Есть и другие данные, например имя хоста машины[1], количество компиляций ядра (одна) и тип процессора – Pentium Pro или выше (на что указывает `i686`). На странице справочного руководства *uname*(1) можно получить дополнительные сведения.

Сборка нового ядра

Ядро Linux – это чудовище со многими щупальцами. Над различными его частями работают много групп людей, и некоторые части программного кода представляют собой пестрое собрание идей, связанных с различными целями проектирования. Однако в целом код ядра ясный и единообразный, и для тех, кто захочет изучить его тайны, это не составит особого труда. Однако, поскольку ведется очень интенсивная разработка ядра, новые выпуски появляются очень часто, иногда ежедневно! Главная причина в том, что почти все драйверы устройств содержатся в ядре, и когда кто-либо обновляет драйвер, необходим выпуск новой версии. Даже несмотря на то, что практически все современные драйверы устройств реализованы в виде загружаемых модулей, тем не менее они распространяются вместе с ядром в виде одного большого пакета.

В настоящее время «официальную» версию ядра поддерживает Линус Торвальдс. Хотя лицензия GPL позволяет любым желающим модифицировать и выпускать новые версии ядра на тех же условиях, поддержка Линусом официального ядра является удобной договоренностью, помогающей сохранять единообразную нумерацию версий и добиваться единого понимания того, о какой версии ядра идет речь. Чтобы исправить ошибку или добавить новую функцию, нужно лишь отправить исправление Линусу (или тому, кто отвечает за соответствующую версию ядра, потому что сам Линус всегда работает с самой свежей версией ядра), и обычно он включает предлагаемые изменения, если они не конфликтуют со всем остальным. Номера версий ядра имеют следующий формат:

```
major.minor.patchlevel
```

major – это основной номер версии, который редко изменяется, *minor* – дополнительный номер версии, указывающий на текущий «штамм» версии ядра[2], а *patch-*

---

[1] На котором выполняется команда *uname*, в данном примере – `pooh`. – *Примеч. науч. ред.*

[2] Принято соглашение, что нечетные номера `minor` присваиваются «неустойчивым» версиям ядра. Когда версия признается успешной, ее код переносится в «устойчивую» версию, у которой `minor` присваиваются четные номера. – *Примеч. науч. ред.*

level – номер патча (заплатки) к текущей версии ядра. Примерами версий ядра являются 2.4.4 (патч уровня 4 ядра версии 2.4) и 2.6.11.4 (подверсия 4 патча уровня 11 ядра версии 2.6).

Ознакомиться с историей развития существующих версий ядра можно на сайте *http://www.kernel.org*.

В вашей системе исходные тексты ядра, скорее всего, находятся в каталоге */usr/src/linux* (если только у вас не дистрибутив Debian, в котором исходные тексты ядра находятся в каталоге */usr/src/kernel-source-versionsnumber*). Если вы собираетесь заново собрать ядро, используя лишь имеющиеся у вас исходные тексты, нет необходимости получать новые файлы или патчи (здесь предполагается, что исходные тексты были установлены во время установки дистрибутива). Если вы хотите обновить ядро, установив его новейшую версию, вам следует воспользоваться указаниями из следующего раздела.

Получение исходных текстов ядра

Официальная версия ядра выпускается в виде архива *tar*, сжатого утилитой *gzip*, в котором содержатся исходные тексты и ряд патчей – по одному файлу для каждого уровня. Архив *tar* содержит версию исходных текстов без патчей. Например, есть *tar*-файл, содержащий исходные тексты ядра версии 2.6.0, на которые не были наложены исправления. Каждый последующий уровень патчей выпускается как патч-файл (созданный с помощью программы *diff*), который можно применить, используя программу *patch*. В разделе «Обновление файлов с помощью patch» главы 21 мы подробно опишем, как пользоваться программой *patch*.

Допустим, вы собираетесь обновить ядро до версии 2.6 с уровнем патчей 4. Вам потребуются исходные тексты версии 2.6 (файл может называться *v2.6.0.tar.gz*) и патчи уровней 1–4. Эти файлы будут называться *patch1*, *patch2* и т. д. (Вам потребуются все эти файлы патчей до того номера версии, который вы хотите в итоге установить. Обычно эти файлы весьма невелики и находятся на архивных сайтах в сжатом виде.) Все файлы можно найти в каталоге *kernel* архивных FTP-сайтов Linux. Например, на сайте *ftp://ftp.kernel.org*, где исходные тексты и патчи для версии 2.6 находятся в каталоге */pub/linux/kernel/v2.6* в виде *tar*-архивов, сжатых с помощью утилиты *gzip* или *bzip2*.

Если у вас уже установлен какой-то уровень патчей ядра (например, версия 2.6, уровень патчей 2) и вы хотите обновить ядро до нового уровня патчей, вы можете просто применить патчи, следующие за версией, которая у вас уже есть, до того уровня, который вам требуется. Если вы обновляете ядро, например с версии 2.6, уровень 2 до версии 2.6, уровень 4, вам понадобятся патч-файлы для 2.6.3 и 2.6.4.

Распаковка исходных текстов

Прежде всего, нужно распаковать архив с исходными текстами из каталога */usr/src*. Для этого выполните следующие команды:

```
rutabaga# cd /usr/src
rutabaga# mv linux linux.old
rutabaga# tar xzf v2.6.0.tar.gz
```

В результате ваше прежнее дерево каталогов с исходными текстами ядра будет сохранено как */usr/src/linux.old*, и будет создано дерево */usr/src/linux*, содер-

жащее новые исходные тексты. Обратите внимание: *tar*-файл с исходными текстами содержит подкаталог *linux*.

Исходные тексты текущей версии ядра нужно хранить в каталоге */usr/src/linux*, поскольку есть две символические ссылки – */usr/include/linux* и */usr/include/asm*, указывающие на дерево с исходными текстами текущего ядра, которые обеспечивают доступ к некоторым заголовочным файлам при компиляции программ. (Исходные тексты ядра всегда должен быть доступны, чтобы можно было использовать заголовочные файлы при компиляции программ.[1]) Если вы хотите сохранить несколько деревьев с исходными текстами ядра, следите за тем, чтобы ссылка */usr/src/linux* указывала на каталог с самой свежей версией.

Применение патчей

Для применения патч-файлов используется программа *patch*. Допустим, имеются сжатые файлы от *patch1.gz* до *patch4.gz*. Они должны применяться из главного каталога с исходными текстами ядра. Это означает не то, что сами они должны там находиться, но что программа *patch* должна запускаться из этого каталога, например */usr/src/linux*. Для каждого патч-файла выполните команду

```
gunzip -c patchfile | patch -p1
```

из */usr/src*. Параметр *-p1* указывает программе *patch*, что она не должна отрезать какую-либо часть имени у файлов, находящихся в патч-файле, за исключением первой.

Все патчи должны применяться в порядке нумерации их уровня, что очень важно. Учтите, что использование символов-заместителей типа *patch** недопустимо, поскольку * использует порядок ASCII, а не числовой. (В противном случае за *patch1* будут следовать *patch10* и *patch11*, а не *patch2* и *patch3*.) Лучше всего вручную выполнить указанную команду последовательно для каждого патч-файла, тогда вы будете уверены в том, что делаете это в правильном порядке.

Проблем при применении патчей к дереву каталогов с исходными текстами возникать не должно, если делать это по порядку и не пытаться дважды применить один и тот же патч. При возникновении проблем посмотрите страницы справочного руководства для утилиты *patch*. Если ничто другое не помогает, удалите новое дерево каталогов с исходными текстами и повторите все сначала из исходного *tar*-файла.

Чтобы еще раз убедиться в том, что все патчи применены успешно, выполните команды:

```
find /usr/src/linux -follow -name "*.rej" -print
find /usr/src/linux -follow -name "*#" -print
```

В результате будут выведены файлы, «отвергнутые» в процессе применения патчей. Если такие файлы существуют, в них будут содержаться те части патч-файлов, которые по какой-либо причине не смогли быть применены. Просмотрите их и при наличии каких-либо сомнений начните все сначала. Нельзя надеяться

[1] Вообще говоря, для целей разработки ПО достаточно хранить только заголовочные файлы (файлы определений *.*h*) кода ядра, что на порядки меньше по объему полных исходных кодов. – *Примеч. науч. ред.*

на то, что ядро правильно скомпилируется или будет работать, если процесс установки патчей прошел неудачно и что-либо было отвергнуто.

Есть удобный сценарий для применения патчей к ядру, который можно найти в файле *scripts/patch-kernel*. Но, как и в других подобных случаях, следует хорошо представлять себе смысл процедуры, осуществляемой с помощью средств автоматизации, в особенности когда она касается самой основы операционной системы – ядра.

Сборка ядра

Сборка ядра производится в шесть этапов, которые должны пройти без труда. На следующих страницах все эти этапы описываются подробно.

1. Установите нужные версии всех необходимых инструментов и утилит. Перечень требований находится в подкаталоге *Documentation/Changes* каталога с исходными текстами ядра.
2. Запустите команду *make config*, при этом вам будут заданы различные вопросы относительно того, какие драйверы вы собираетесь включить в ядро. Можно также воспользоваться более удобными вариантами *make menuconfig* или *make xconfig* (если запущено окружение рабочего стола KDE) либо *make gconfig* (если запущено окружение рабочего стола GNOME).

   Если вы уже выполняли ранее компиляцию ядра, а потом применяли патчи к новой версии, можно запустить команду *make oldconfig*, чтобы использовать старый файл настройки, но получить приглашение для ввода новых параметров, отсутствовавших в прежнем ядре.
3. Запустите команду *make dep*, чтобы собрать сведения о зависимостях для каждого исходного файла и включить их в различные *make*-файлы. Для текущей версии ядра (серии 2.6) этот шаг можно пропустить.
4. Если раньше вы уже собирали ядро из этого дерева каталогов с исходными текстами, выполните команду *make clean*, чтобы удалить прежние объектные файлы и инициировать полную повторную сборку ядра.
5. Запустите команду *make bzImage*, чтобы построить само ядро.
6. Сходите выпить чашечку кофе (или пару чашечек, в зависимости от быстродействия вашей машины и объема оперативной памяти).
7. Установите новый образ ядра на загрузочную дискету или с помощью GRUB. Поместить ядро на загрузочную дискету можно с помощью *make bzDisk*.

Все эти команды нужно выполнять из каталога */usr/src/linux*, за исключением шага 5, который можно выполнить в любом месте.

В исходных текстах ядра имеется файл *README*, который должен располагаться в каталоге */usr/src/linux/README* вашей системы. Прочтите его. В нем содержатся самые последние сведения о компиляции ядра, которые могут быть более свежими, чем изложенные здесь. Следуйте описанным там шагам, используя в качестве справочника описания, приведенные далее в этом разделе. Если исходные тексты ядра были установлены из пакета, входящего в состав дистрибутива, в дереве каталогов с исходными текстами может также находиться файл с примечаниями, относящимися к данному дистрибутиву, о параметрах настройки ядра и возможных изменениях, внесенных в исходные тексты, которые можно загрузить из сети.

Настройка ядра: make config

На первом этапе нужно выполнить *make config*. При этом запускается сценарий, задающий ряд вопросов относительно того, какие драйверы должны быть включены в ядро, на которые нужно отвечать «yes» или «no». Для каждого вопроса установлен ответ по умолчанию, но будьте внимательны: ответы по умолчанию могут не соответствовать тому, что вам требуется. (Если имеется несколько вариантов ответа, вариант по умолчанию будет показан заглавной буквой, например `[Y/n]`.) Ваши ответы станут ответами по умолчанию, когда вы в следующий раз будете собирать ядро из этого дерева с исходными текстами.

Отвечайте на каждый вопрос, нажимая клавишу Enter при выборе ответа по умолчанию либо вводя `y` или `n`, а затем нажимая Enter. Не все вопросы предполагают ответ типа yes/no, иногда вас могут попросить ввести число или какое-либо другое значение. Некоторые вопросы о тех или иных параметрах настройки допускают помимо `y` или `n` ответ `m`. Такой ответ позволяет скомпилировать соответствующую функцию ядра в виде загружаемого модуля ядра, а не встраивать ее непосредственно в образ ядра. Загружаемые модули, о которых говорится в следующем разделе, «Загружаемые драйверы устройств», позволяют частям ядра (например, драйверам устройств) загружаться и выгружаться во время работы системы по мере необходимости. Если вы не уверены в том, какой ответ выбрать, введите в ответ на приглашение символ `?`. В этом случае для большинства параметров будет выведено сообщение с некоторыми сведениями о настраиваемом параметре.

Система запоминает выбранные параметры настройки каждый раз, когда запускается команда *make config*, что позволяет быстро находить и изменять только требуемые параметры, исключая необходимость повторного ввода всех настраиваемых параметров.

Некоторые считают, что *make config* требует сейчас столько параметров, что работать с ней вручную стало слишком сложно, поскольку приходится тратить много усилий для выбора правильного ответа на задаваемые вопросы. По этой причине многие переходят на альтернативные варианты, описываемые ниже.

Альтернативой запуску *make config* служит *make xconfig*, которая компилирует и запускает программу настройки ядра в системе X Window. Для использования этой возможности у вас должна быть запущена система X Window, кроме того, должны быть установлены соответствующие библиотеки X11, Qt и т. д. Вместо ответов на серию вопросов утилита настройки, работающая в системе X, требует пометить флажками те параметры ядра, которые вы хотите включить.

Кроме того, можно воспользоваться командой *make menuconfig*, которая использует в текстовом режиме библиотеку *curses* и предоставляет аналогичную основанную на меню возможность настройки ядра, если у вас не установлена система X. Пользоваться командами *make menuconfig* и *make xconfig* значительно удобнее, чем *make config*, особенно потому, что можно вернуться к некоторому параметру и изменить его, прежде чем сохранить настройки. Однако здесь мы опишем процесс настройки в линейном порядке, как это делает команда *make config*.

Ниже приведена часть сеанса работы с *make config*. При работе с *make menuconfig* или *make xconfig* вы столкнетесь с тем же набором параметров, только представленным в более дружественном пользователю виде (мы настоятельно реко-

мендуем по возможности пользоваться этими инструментами, поскольку в многочисленных параметрах конфигурации легко запутаться):

```
pooh:/usr/src/linux # make config
scripts/kconfig/conf arch/i386/Kconfig
#
# using defaults found in .config
#
*
* Linux Kernel Configuration
*
*
* Code maturity level options
*
Prompt for development and/or incomplete code/drivers (EXPERIMENTAL) [Y/n/?]
Select only drivers expected to compile cleanly (CLEAN_COMPILE) [Y/n/?]
*
* General setup
*
Local version - append to kernel release (LOCALVERSION) [-default]
Support for paging of anonymous memory (swap) (SWAP) [Y/n/?]
System V IPC (SYSVIPC) [Y/n/?]
POSIX Message Queues (POSIX_MQUEUE) [Y/n/?]
BSD Process Accounting (BSD_PROCESS_ACCT) [Y/n/?]
BSD Process Accounting version 3 file format (BSD_PROCESS_ACCT_V3) [Y/n/?]
Sysctl support (SYSCTL) [Y/n/?]
Auditing support (AUDIT) [Y/n/?]
Enable system-call auditing support (AUDITSYSCALL) [Y/n/?]
Kernel log buffer size (16 => 64KB, 17 => 128KB) (LOG_BUF_SHIFT) [17]
Support for hot-pluggable devices (HOTPLUG) [Y/?] y
Kernel Userspace Events (KOBJECT_UEVENT) [Y/n/?]
Kernel .config support (IKCONFIG) [Y/n/?]
Enable access to .config through /proc/config.gz (IKCONFIG_PROC) [Y/n/?]
*
* Configure standard kernel features (for small systems)
*
Configure standard kernel features (for small systems) (EMBEDDED) [N/y/?]
Load all symbols for debugging/kksymoops (KALLSYMS) [Y/?] (NEW) y
Include all symbols in kallsyms (KALLSYMS_ALL) [N/y/?]
Do an extra kallsyms pass (KALLSYMS_EXTRA_PASS) [N/y/?]
*
* Loadable module support
*
Enable loadable module support (MODULES) [Y/n/?]
Module unloading (MODULE_UNLOAD) [Y/n/?]
Forced module unloading (MODULE_FORCE_UNLOAD) [Y/n/?]
Module versioning support (EXPERIMENTAL) (MODVERSIONS) [Y/n/?]
Source checksum for all modules (MODULE_SRCVERSION_ALL) [Y/n/?]
Automatic kernel module loading (KMOD) [Y/n/?]
*
* Processor type and features
*
Subarchitecture Type
> 1. PC-compatible (X86_PC)
```

```
2. AMD Elan (X86_ELAN)
3. Voyager (NCR) (X86_VOYAGER)
4. NUMAQ (IBM/Sequent) (X86_NUMAQ)
5. SGI 320/540 (Visual Workstation) (X86_VISWS)
choice[1-5]:
Processor family
1. 386 (M386)
2. 486 (M486)
> 3. 586/K5/5x86/6x86/6x86MX (M586)
4. Pentium-Classic (M586TSC)
5. Pentium-MMX (M586MMX)
6. Pentium-Pro (M686)
7. Pentium-II/Celeron(pre-Coppermine) (MPENTIUMII)
8. Pentium-III/Celeron(Coppermine)/Pentium-III Xeon (MPENTIUMIII)
9. Pentium M (MPENTIUMM)
10. Pentium-4/Celeron(P4-based)/Pentium-4 M/Xeon (MPENTIUM4)
11. K6/K6-II/K6-III (MK6)
12. Athlon/Duron/K7 (MK7)
13. Opteron/Athlon64/Hammer/K8 (MK8)
14. Crusoe (MCRUSOE)
15. Efficeon (MEFFICEON)
16. Winchip-C6 (MWINCHIPC6)
17. Winchip-2 (MWINCHIP2)
18. Winchip-2A/Winchip-3 (MWINCHIP3D)
19. CyrixIII/VIA-C3 (MCYRIXIII)
20. VIA C3-2 (Nehemiah) (MVIAC3_2)
choice[1-20]:
Generic x86 support (X86_GENERIC) [Y/n/?]
HPET Timer Support (HPET_TIMER) [N/y/?]
Symmetric multi-processing support (SMP) [N/y/?]
Preemptible Kernel (PREEMPT) [N/y/?]
Local APIC support on uniprocessors (X86_UP_APIC) [Y/n/?]
IO-APIC support on uniprocessors (X86_UP_IOAPIC) [Y/n/?]
Disable local/IO APIC by default (X86_APIC_OFF) [Y/n/?]
Machine Check Exception (X86_MCE) [Y/n/?]
Check for non-fatal errors on AMD Athlon/Duron / Intel Pentium 4 (X86_MCE_NONFATAL)
[N/m/y/?]
check for P4 thermal throttling interrupt. (X86_MCE_P4THERMAL) [Y/n/?]
Toshiba Laptop support (TOSHIBA) [M/n/y/?]

...и так далее...
*** End of Linux kernel configuration.
*** Check the top-level Makefile for additional configuration.
*** Next, you may run 'make bzImage', 'make bzdisk', or 'make install'.
```

Если вы сохранили информацию о том, какие аппаратные устройства установлены на вашем компьютере, во время установки Linux, то ее может оказаться достаточно для того, чтобы ответить на вопросы относительно параметров настройки, которые в большинстве своем должны быть просты. Если вам не знакома какая-то функция, значит, это специальная функция, которая вам не нужна.

Заметим, что не все драйверы устройств Linux реально встраиваются в ядро. Некоторые драйверы имеются только в виде загружаемых модулей, распространяемых отдельно от исходных текстов ядра. (Как уже говорилось, некоторые драй-

веры можно либо встраивать в ядро, либо компилировать как модули. В других случаях может быть доступен только один из вариантов.)

Если вы не можете найти свое любимое устройство в списке, представленном *make config*, вполне возможно, что его драйвер существует в виде модуля или отдельного патча ядра. Поищите по сайтам FTP и архивным CD-ROM, если не удастся отыскать то, что требуется. В разделе «Загружаемые драйверы устройств» далее в этой главе подробно описываются модули ядра.

Вопросы, описания которых приводятся ниже, задаются в процессе настройки ядра версии 2.6.11.4. Если на исходные тексты накладывались дополнительные патчи, это могло привести к появлению новых вопросов. То же самое относится к более поздним версиям ядра. Обратите внимание: в следующем списке присутствуют далеко не все параметры настройки ядра, просто их слишком много, причем ответы на большинство из них достаточно очевидны. Мы выделили те из них, которые действительно требуют некоторого пояснения. Помните, если вы не уверены, как ответить на тот или иной вопрос, выбирайте ответ по умолчанию, так как чаще всего это лучший выбор. Кроме того, в случае сомнений можно ввести символ ? и ознакомиться со справочной информацией.

Следующие пункты представляют собой параметры верхнего уровня и могут содержать вложенные пункты.

Prompt for development and/or incomplete code/drivers

Ответьте «yes», если вы хотите использовать новые функции, которые разработчики пока считают недостаточно стабильными. Этот параметр вам не нужен, если вы не собираетесь принимать участие в тестировании новых особенностей.

System V IPC

Ответ «yes» включает поддержку ядром функций взаимодействия между процессами (IPC) для System V, таких как *msgrcv* и *msgsnd*. Это требуется для некоторых программ, перенесенных с System V. Следует ответить «yes», если только у вас нет сильного отвращения к этим функциям.

Sysctl support

Этот параметр указывает, что ядро должно позволять менять свои параметры «на лету» без перезагрузки. Полезно включить ее, если только вы не настолько ограничены в памяти, что не выдержите увеличения ядра на 8 Кбайт, вызываемого этим параметром.

Enable loadable module support

Включает поддержку динамически загружаемых дополнительных модулей. Вам наверняка нужно активировать ее.

Module versioning support

Специальный параметр, делающий возможным использование с одной версией ядра модулей, компилированных с другой версией ядра. С этим связан ряд проблем. Ответьте «no», если нет полной уверенности в том, что вы делаете.

Automatic kernel module loading

При включении этого параметра ядро может автоматически загружать и выгружать динамически загружаемые модули по мере необходимости.

Processor family (предоставляется возможность выбора 20 различных типов процессоров) [586/K5/5x86/6x86/6x86MX]

Здесь нужно указать тип установленного у вас центрального процессора. В результате ядро компилируется с оптимизацией, специально настроенной на вашу машину. Учтите, что, если вы зададите более высокий тип процессора, чем тот, который у вас есть в действительности, ядро может оказаться неработоспособным. Напротив, при выборе более раннего типа (например, выбор типа Pentium, когда в компьютере фактически установлен Pentium IV) работоспособность ядра сохраняется всегда, но при этом его быстродействие может оказаться более низким, чем при указании более совершенной модели.

Symmetric multi'processing support

Включает поддержку ядром более одного процессора. Если у вас установлено несколько процессоров, ответьте «yes», в противном случае – «no».

Power management options (ACPI, APM)

Целый раздел параметров управления питанием, предназначенных главным образом для ноутбуков, однако такие параметры, как *suspend-to-disk* ("*hibernation*"), могут с успехом использоваться на обычных рабочих станциях. Тем не менее некорректная установка параметров управления питанием может приводить к определенным проблемам, включая полный отказ во время загрузки. При появлении подобных проблем попробуйте пересобрать ядро без возможности управления питанием или передавайте ядру во время загрузки параметры командной строки `noapm` и `noacpi`, чтобы отключить функции управления питанием.

PCI support

Включите этот параметр, если на материнской плате имеется шина PCI и в системе установлены устройства PCI. Для обнаружения и активации PCI-устройств используется PCI BIOS. Поддержка PCI BIOS ядром необходима для использования системой каких-либо PCI-устройств.

Parallel port support

Включите этот параметр, если у вас есть параллельный порт и вам необходим доступ к нему из Linux. Linux может использовать параллельный порт не только для подключения принтеров, но и для PLIP (сетевой протокол для параллельных линий), накопителей ZIP, сканеров и других устройств. В большинстве случаев для подключения устройства к параллельному порту нужен дополнительный драйвер. Если у вас имеется современный принтер, поддерживающий возможность двустороннего обмена информацией с компьютером, необходимо будет также включить параметр `IEEE 1284 transfer modes`.

Normal floppy disk support

Ответьте «yes», если только вы не желаете, чтобы гибкие диски не поддерживались (это поможет сберечь немного памяти в системах, где не нужна поддержка гибких дисков). Если у вас имеется всего один привод гибких дисков на ноутбуке серии IBM Thinkpad, тогда необходимо будет во время загрузки передать параметр `floppy=thinkpad`.

Parallel port IDE device support

Этот параметр включает поддержку IDE-устройств, подключаемых к параллельному порту, таких как переносные приводы CD-ROM.

Packet writing on CD/DVD media

Если в вашем распоряжении имеется современное устройство записи дисков CD или DVD, выбором данного параметра можно разрешить пакетный режим записи (в противовес режиму записи по дорожкам).

Enhanced IDE/MFM/RLL disk/cdrom/tape/floppy support

Ответьте «yes», если вам нужна поддержка привода IDE/MFM/RLL. После ответа «yes» будет предложено указать тип используемого устройства (жесткие диски, приводы компакт-дисков, ленточные накопители и приводы гибких дисков), доступ к которому будет осуществляться через интерфейс IDE. Если у вас нет приводов IDE (только SCSI), безопаснее будет оставить этот параметр выключенным.

SCSI support

Если у вас есть хотя бы один контроллер SCSI, отвечайте «yes». Вам будет задан ряд вопросов относительно конкретных SCSI-устройств, имеющихся в машине. Вам необходимо знать, какие аппаратные устройства у вас установлены. Все эти вопросы касаются конкретных типов чипов и плат SCSI-контроллеров, которые у вас установлены. Если вы не уверены в том, какой тип SCSI-контроллера есть у вас, посмотрите документацию по своим устройствам или загляните в документы «Linux HOWTO».

Вам также будет задан вопрос о необходимости поддержки дисков, ленточных накопителей, приводов CD-ROM и других устройств SCSI. Задайте те параметры, которые соответствуют имеющимся у вас устройствам.

Если устройств SCSI у вас нет, нужно ответить «no», в результате чего размер ядра существенно уменьшится.

Old CD'ROM drivers

Ряд вопросов относительно специфических драйверов CD-ROM, поддерживаемых ядром, таких как Sony CDU31A/33A, Mitsumi или SoundBlaster Pro. Если у вас есть контроллер CD-ROM типа SCSI или IDE (и вы уже выбрали его поддержку ранее), этот параметр выбирать не нужно. У некоторых приводов CD-ROM есть специальные интерфейсные платы, и эти параметры подключают драйверы для таких плат.

Networking support

Ответьте «yes», если требуется любая поддержка ядром работы в сети (в том числе TCP/IP, SLIP, PPP, NFS и т. п.).

Networking options

Если ранее вы выбрали поддержку сети, вам будет задан ряд вопросов относительно того, какие сетевые возможности вы хотите видеть в ядре. Если только у вас нет особых требований к работе в сети (и тогда вы знаете правильные ответы на вопросы), достаточно ввести значения по умолчанию. Ряд вопросов понятен только посвященным (такие как `PF_KEY sockets`), и почти всегда в качестве ответа на них следует выбирать значения по умолчанию.

Network device support

Задается ряд вопросов о конкретных сетевых контроллерах, поддерживаемых Linux. Если вы собираетесь использовать карту Ethernet (или другой сетевой контроллер), включите те параметры, которые соответствуют вашему устройству. Как и в случае устройств SCSI, следует обратиться к имеющимся для ваших устройств руководствам или к документам «Linux HOWTO» (например, «Ethernet HOWTO»), чтобы определить, какой драйвер подходит для вашего сетевого контроллера.

Amateur Radio support

Этот параметр включает базовую поддержку сетевого взаимодействия через общедоступные радиочастоты. Если у вас есть соответствующее оборудование, включите этот параметр и прочтите HOWTO по AX25 и HAM.

ISDN subsystem

Если у вас установлено оборудование ISDN, включите этот параметр и выберите соответствующий вашему устройству драйвер ISDN. Скорее всего, вам понадобится также выбрать параметр `Support synchronous PPP` (см. раздел «PPP поверх ISDN» главы 13). В настоящее время Linux осуществляет переход со старой, так называемой ISDN4Linux, модели поддержки на новый стандарт CAPI 2.0. Пока поддержка обоих стандартов должна корректно работать с большинством устройств ISDN, однако в будущем предполагается полный переход на стандарт CAPI 2.0.

Telephony support

Linux поддерживает некоторые интерфейсные карты, позволяющие подключать обычный телефонный аппарат для использования интернет-телефонии через протокол VoIP (передача голосовой информации по протоколу IP). Как оговаривается в документации, данный параметр не имеет никакого отношения к модемам, таким образом, включать данный параметр следует, только если у вас имеется интерфейсная карта.

Character devices

Linux поддерживает несколько особых «символьных» устройств, таких как контроллеры последовательных и параллельных портов, ленточные накопители и мыши с особыми интерфейсами (не те, которые подключаются к последовательному порту или порту USB, как мышь Microsoft).

Sound

В этом разделе будет предложено выбрать одну из двух звуковых подсистем: более новую ALSA (Advanced Linux Sound Architecture – расширенная архитектура воспроизведения звука в Linux) или старую OSS (Open Sound System – открытая звуковая система). Лучше отдать предпочтение подсистеме ALSA, если она поддерживает имеющиеся звуковые устройства.

USB support

Установка этого параметра позволит получить поддержку многих устройств USB. В частности, если предполагается использовать популярные ныне модули флэш-памяти, необходимо будет дополнительно включить параметр `USB Mass Storage Support`. Этот же параметр затрагивает поддержку цифровых фото- и видеокамер, подключаемых к компьютеру через порт USB.

Filesystems

Ряд вопросов по файловым системам, поддерживаемым ядром. Как было отмечено в разделе «Управление файловыми системами» главы 10, система поддерживает несколько типов файловых систем, и вы можете выбрать те, которые следует включить в ядро. Практически всегда нужно включать поддержку второй расширенной файловой системы и файловой системы */proc*. Если вы хотите иметь непосредственный доступ из Linux к файлам MS-DOS, нужно включить поддержку файловой системы MS-DOS, а кроме того, желательно добавить поддержку файловой системы ISO 9660 для доступа к файлам на CD-ROM (чаще всего на CD-ROM используется эта система).

Kernel hacking

В этом разделе содержатся параметры, имеющие смысл, если вы собираетесь сами заняться разработкой ядра. Если не собираетесь, ответьте «no».

Подготовка к компиляции: make clean

Если вы хотите, чтобы ядро было полностью перекомпилировано, то в этом месте нужно выполнить команду *make clean*. Эта команда удалит из дерева исходного кода все объектные файлы, оставшиеся от предыдущей компиляции. Если вы ни разу не собирали ядро из этого дерева, то вполне можете пропустить этот шаг (хотя он и не повредит). Если вы произвели в ядре небольшие изменения, то этот шаг можно пропустить, чтобы перекомпилировались только изменившиеся файлы. Команда *make clean* всего лишь гарантирует, что все ядро будет перекомпилировано «с нуля», и если у вас есть какие-то сомнения, выполните для верности эту команду.

Компиляция ядра

Теперь можно компилировать ядро. Это делается командой *make bzImage*. Лучше всего собирать ядро, когда система не сильно загружена и много свободной памяти, которая может использоваться для компиляции. Если к машине подключены другие пользователи или вы сами запустили большие приложения, например X Window System или другой процесс компиляции, то сборка может идти очень медленно. Основным фактором является память. Если памяти мало и системе приходится осуществлять свопинг, компиляция происходит медленно, даже если процессор мощный.

Продолжительность компиляции ядра может быть самой разной – от нескольких минут до многих часов, в зависимости от того, какая у вас машина. Это не должно вызывать удивление, поскольку объем исходных текстов ядра превышает 80 Мбайт. На медленных машинах с 16 Мбайт памяти или менее для полной сборки ядра может потребоваться несколько часов. На более быстрых машинах с большим объемом памяти можно уложиться менее чем в полчаса. В вашем конкретном случае это время может сильно отличаться.

Если во время компиляции появляются сообщения об ошибках или предупреждения, то не следует рассчитывать, что получившееся ядро будет нормально работать. Обычно сборка прекращается при возникновении ошибки. Ошибки могут быть связаны с неправильным применением патчей, проблемами на этапе *make config* или действительными ошибками в исходных текстах. В «базовом» ядре последнее случается редко, а вот если вы работаете с разрабатываемой вер-

сией или тестируете новые драйверы, такое вполне может произойти. Если имеются сомнения, удалите все дерево каталогов с исходными текстами ядра и начните все сначала.

По завершении компиляции вы получите файл *bzImage* в каталоге */usr/src/linux/arch/i386/boot*. (Если вы собираете Linux на платформе, отличной от Intel x86, то образ ядра будет в другом подкаталоге внутри каталога *arch*.) Такое название файла объясняется тем, что это исполняемый образ ядра, в котором используется сжатие с помощью алгоритма *bzip2*. При загрузке ядро само разархивирует себя в память. Не пытайтесь применять *bzip2* или *bunzip2* к *bzImage*! Благодаря такому сжатию ядро занимает значительно меньше дискового пространства и может поместиться на дискету. Раньше ядро поддерживало оба алгоритма сжатия – *gzip* и *bzip2*, причем первый создавал файл с именем *zImage*. Поскольку *bzImage* имеет большую степень сжатия, не следует пользоваться *gzip*, так как получающиеся файлы для ядра современных версий оказываются слишком велики, чтобы их можно было установить.

Если включить в ядро слишком много функций, в конце компиляции можно получить сообщение об ошибке `kernel too big` (слишком большое ядро). Это случается редко, поскольку одной машине требуется поддержка лишь ограниченного количества аппаратных средств, но все же может произойти. В такой ситуации остается только один выход: скомпилировать некоторые функции ядра в виде модулей (подробности приведены в разделе «Загружаемые драйверы устройств»).

Теперь нужно выполнить *rdev* с новым образом ядра, чтобы убедиться в правильной установке устройства корневой файловой системы, режима консоли SVGA и других параметров. Эта процедура описана в разделе «Использование загрузочной дискеты» главы 17.

Установка ядра

Имея в руках новое ядро, можно настроить его начальную загрузку. Для этого нужно поместить образ ядра на дискету или настроить GRUB для загрузки ядра с жесткого диска. Эти вопросы обсуждались в разделе «Использование загрузочной дискеты» главы 17. Для работы с новым ядром установите один из описанных способов его начальной загрузки и перезагрузите систему.

Всегда сохраняйте заведомо работоспособное ядро, которое можно загрузить. Либо обеспечьте возможность выбора сохраненного предшествующего ядра в GRUB, либо проверяйте новое ядро, загружаясь сначала с дискеты. Это выручит вас в случае, если вы по ошибке не включите в новое ядро критически важный драйвер, сделав тем самым невозможной загрузку системы.

Загружаемые драйверы устройств

Обычно драйверы устройств непосредственно включаются в состав ядра. Это делается по нескольким причинам. Во-первых, почти всем драйверам устройств требуется особый доступ к аппаратным устройствам, который обеспечивается, если драйвер является частью кода ядра. Такой доступ к устройствам нелегко получить пользовательской программе. Во-вторых, драйверы значительно легче

реализуются при включении их в состав ядра: они получают полный доступ к структурам данных и другим находящимся в ядре функциям, к которым они могут свободно обращаться.

С таким конгломерированным ядром, содержащим в себе все драйверы, возникает ряд проблем. Во-первых, для избирательного включения драйверов устройств в ядро необходимо, чтобы администратор заново скомпилировал ядро, как было показано в предыдущем разделе. Во-вторых, при таком механизме разработчики драйверов могут писать неряшливый программный код: ничто не мешает программисту написать не вполне модульный программный код, который, например, напрямую обращается к данным, принадлежащим другим частям ядра. Эта проблема осложняется кооперативным характером разработки ядра Linux, из-за чего не во всех частях кода соблюдается надлежащая сдержанность. Это осложняет сопровождение и отладку.

В попытке уйти от этой парадигмы в ядро Linux была включена поддержка загружаемых драйверов устройств – таких, которые с помощью нескольких команд можно загрузить в память или удалить из нее во время работы системы. Такие драйверы продолжают оставаться частью ядра, но они компилируются отдельно и включаются только после загрузки. Загружаемые драйверы устройств, или *модули*, обычно загружаются в память с помощью команд, помещаемых в один из загрузочных *rc*-сценариев.

Модули обеспечивают более ясный интерфейс для написания драйверов. В известной мере они требуют модульности кода и следования определенным правилам программирования. (Это не мешает, впрочем, программисту нарушить соглашение и написать немодульный код. После того как модуль загружен, он может сеять беспорядок столь же свободно, как если бы компилировался непосредственно в ядре.) Благодаря использованию модулей становится легче отлаживать драйверы: можно выгрузить модуль, перекомпилировать его и заново загрузить без необходимости перезагружать систему или перекомпилировать все ядро. Модули можно использовать не только для драйверов устройств, но и для других частей системы, например файловых систем.

Большинство драйверов устройств и многие другие функции ядра реализованы в Linux в виде модулей. Одним из таких модулей является драйвер параллельного порта PC (или *parport_pc*) для устройств, подключаемых к параллельному порту (кроме того, существуют дополнительные драйверы таких устройств, как принтеры, обладающие возможностью двустороннего обмена информацией с компьютером). Если вы собираетесь использовать этот драйвер в своей системе, неплохо разобраться, как компилировать, загружать и выгружать модули. Хотя ничто не мешает скомпилировать этот модуль статически в составе ядра, драйвер параллельного порта используется слишком редко (только когда необходимо распечатать документ на принтере, подключенном непосредственно к компьютеру, что случается один-два раза в день), чтобы позволить драйверу постоянно занимать драгоценную память. За информацией о подключении принтеров к параллельному порту обращайтесь к документу «Linux Printing HOWTO».

Установка ядра

Теперь мы поговорим о том, как загружать и выгружать модули ядра. Прежде всего, вам понадобится пакет *module-init-tools*, в котором содержатся команды,

используемые для загрузки и выгрузки модулей из ядра. На архивных сайтах FTP его обычно можно найти как *module-init-tools-versionnumber.tar.bz2* в каталоге с исходными текстами ядра. В этом пакете содержатся исходные тексты команд *insmod*, *modprobe*, *rmmod* и *lsmod*. Эти команды включаются в большинство дистрибутивов Linux и помещаются в каталог */sbin*. Если эти команды у вас уже установлены, вероятно, загружать пакет *module-init-tools* не требуется. Однако если вы загрузите пакет и заново скомпилируете эти команды, то можете быть уверены в том, что у вас стоит их самая последняя версия.

Чтобы заново скомпилировать эти команды, распакуйте *module-init-tools-versionnumber.tar.bz2* (скажем, в подкаталог */usr/src*). Следуйте инструкции по установке, которая там находится. Обычно все, что требуется, – это выполнить команды *make* и *make install* (в качестве суперпользователя). После этого нужные команды установятся в каталог */sbin* и будут готовы к использованию.

Компиляция модулей

Модуль является просто отдельным объектным файлом, содержащим весь программный код драйвера. Модуль *parport_pc*, например, может называться *parport_pc.ko*. Во многих системах сами модули хранятся в дереве каталогов в */lib/modules/kernelversion*, где находятся каталоги, соответствующие различным типам модулей. Например, модули, скомпилированные для ядра 2.6.8, будут находиться в каталоге */lib/modules/2.6.8*. Возможно, какие-то модули уже установлены в вашей системе; проверьте соответствующий каталог. Обратите внимание: файлы модулей ядра, в отличие от других объектных файлов, имеют расширение *.ko*, чтобы показать, что они являются модулями ядра. Если вы пользуетесь более старой версией Linux, файлы модулей ядра могут иметь расширение *.o*.

Модули могут входить в состав исходных текстов ядра или находиться вне его. К числу первых относятся наиболее часто используемые драйверы устройств, файловые системы и другие функции, поддерживаемые как часть официального исходного кода ядра. Использовать такие модули просто: во время выполнения *make config*, *make menuconfig* или *make xconfig* выберите сборку некоторой функции в виде модуля. Это нужно сделать для всех функций, которые требуется скомпилировать в виде модулей. Затем, после этапа *make bzImage*, выполните команды:

```
# make modules
# make modules_install
```

В результате модули будут скомпилированы и установлены в каталог */lib/modules/kernelversion*.

Новые модули, которые еще не были интегрированы в официальные исходные тексты ядра или просто слишком экзотичны для этого (например, драйвер какого-нибудь заказного устройства, отсутствующего в свободной продаже), могут поставляться как отдельные внешние модули. Распакуйте архив с таким модулем, откомпилируйте согласно инструкции, которая, надо надеяться, входит в архив, и скопируйте полученный файл в соответствующий подкаталог каталога */lib/modules/kernelversion*. Для некоторых модулей может поставляться сценарий установки, либо на последнем этапе нужно выполнить команду *make install*.

Загрузка модуля

Получив скомпилированный модуль (из исходного кода ядра или внешнего источника), можно загрузить его с помощью команды:

```
insmod module
```

где *module* – это имя объектного файла модуля. Например, команда:

```
insmod /lib/modules/2.6.11.4/kernel/drivers/parport/parport_pc.ko
```

загрузит модуль *parport_pc*, если он будет найден в этом файле.

После загрузки модуль может вывести некоторые данные на консоль (а также в системные журналы), указывающие на то, что он инициализирован. Например, драйвер *parport_pc* может вывести следующее:

```
Jul 26 13:08:41 tigger kernel: pnp: Device 00:09 activated.
Jul 26 13:08:41 tigger kernel: parport: PnPBIOS parport detected.
Jul 26 13:08:41 tigger kernel: parport0: PC-style at 0x378, irq 7 [PCSPP,TRISTATE]
```

Разумеется, точный вид сообщений зависит от модуля. Каждый модуль должен сопровождаться подробной документацией, описывающей, что он делает и как его отлаживать в случае появления проблем.

Вполне возможно, что *insmod* сообщит о невозможности загрузки модуля в ядро ввиду `symbols missing` (отсутствуют символы). Это означает, что модулю, который вы хотите загрузить, требуются некоторые функции, находящиеся в другой части ядра, которые не скомпилированы в ядре и не были загружены в какомлибо другом модуле. В частности, модуль *parport_pc*, используемый в качестве примера, зависит от модуля *parport*, который реализует базовые функции работы с параллельным портом. Нужно постараться узнать, в каком модуле содержатся эти функции, и загрузить с помощью *insmod* сначала этот модуль, а потом другой. В конце концов, вам удастся добиться успеха, хотя и не без некоторого труда, и Linux не была бы верна себе, если бы не предоставила способа получше.

Для начала вам нужно иметь базу данных модулей в файле */lib/modules/kernel-version/modules.dep*. Создать ее можно, выполнив команду:

```
depmod -a
```

Эта команда просматривает все имеющиеся у вас модули и регистрирует те случаи, когда им требуются другие модули. Когда такая база данных есть, вы можете вместо *insmod* воспользоваться командой *modprobe*, которая сверяется с базой данных модулей и при необходимости загружает те модули, которые должны быть загружены раньше требуемого. Например, в нашем файле *modules.dep* среди прочих содержится строка:

```
/lib/modules/2.6.8/kernel/drivers/isdn/i4l/isdn.ko:
/lib/modules/2.6.8/kernel/drivers/net/slhc.ko
```

Она означает, что для загрузки модуля *isdn* (драйвер поддержки ISDN) сначала должен быть загружен модуль *slhc* (содержащий одну из реализаций протокола ISDN). Теперь, если мы загружаем модуль *isdn* командой *modprobe* (этот пример несколько упрощен, поскольку модулю *isdn* требуются дополнительные параметры):

```
modprobe hisax
```

то *modprobe* проверит зависимости и загрузит модуль *slhc*. Если вы скомпилировали модуль для текущего ядра, то нужно сначала выполнить команду *depmod -a*, чтобы *modprobe* могла его найти.

Некоторым модулям требуются так называемые параметры модуля. Например, драйверу может понадобиться номер прерывания (IRQ). Эти параметры можно передать в виде *parametername=parametervalue* как для *insmod*, так и для *modprobe*. В следующем примере модулю *hisax* передаются несколько параметров:

```
tigger # modprobe hisax type=3 protocol=2 io=0x280 irq=10
```

В документации к модулю должно быть сказано, какие параметры он поддерживает. Если вам лень читать документацию, можно воспользоваться умной утилитой *modinfo*, которая, помимо всего прочего, сообщает, какие параметры принимает модуль.

Следует предупредить пользователей дистрибутива Debian относительно модулей. Debian использует файл */etc/modules*, содержащий список всех модулей, которые необходимо загрузить во время начальной загрузки. Если модуль, который вам не нужен, продолжает загружаться, проверьте, не находится ли он в этом списке.

Список загруженных драйверов можно получить командой *lsmod*, например:

```
rutabaga$ lsmod
Module Size Used by
parport 40392 1 parport_pc
```

Выводится также объем используемой модулем памяти в байтах. Таким образом, драйвер *parport* использует здесь 40 Кбайт памяти. Если от этого модуля зависят другие модули, они показываются в третьей колонке.

Модуль можно выгрузить из памяти командой *rmmod*. Например:

```
rmmod parport_pc
```

Аргументом *rmmod* является имя драйвера в том виде, как его выводит команда *lsmod*.

Если вы удовлетворены работой загруженных модулей, можно включить соответствующие команды *insmod* в один из *rc*-сценариев, выполняемых во время начальной загрузки системы. В некоторых дистрибутивах в одном из *rc*-сценариев может быть уже зарезервировано место для команд *insmod*.

Одной из особенностей поддержки модулей является необходимость перекомпиляции модулей при обновлении ядра до нового номера версии или уровня патчей. (Если вы перекомпилируете ядро с сохранением номера версии, этого делать не требуется.) Это делается для того, чтобы обеспечить совместимость модуля с используемой вами версией ядра. Если вы попытаетесь загрузить модуль с более новой или более старой версией ядра в сравнении с той, с которой он компилировался, *insmod* откажется загружать модуль. При компиляции модуля должно работать ядро той версии, с которой он должен исполняться. Поэтому при обновлении ядра сначала обновите ядро и перезагрузите машину с новым ядром, а затем откомпилируйте и загрузите модули. Есть вариант, позволяющий сохранять модули при переходе на другое ядро, но с ним связан ряд проблем, и мы не рекомендуем его использовать.

Автоматическая загрузка модулей

Автоматическая загрузка модулей – очень удобная функция ядра, которая реализована в компоненте ядра с именем *kmod*. С его помощью ядро может автоматически загружать требуемые драйверы устройств и другие модули без вмешательства системного администратора. Если в течение некоторого времени (60 секунд) модули не используются, они автоматически выгружаются.

Чтобы воспользоваться *kmod*, необходимо во время настройки ядра включить его поддержку (`Automatic kernel module loading`) в разделе `Loadable module support`.

Модули, для работы которых нужны другие модули, должны быть перечислены в файле */lib/modules/kernelversion/modules.dep*, а кроме того, необходимо иметь псевдонимы для основного и дополнительного номеров в файле */etc/conf.modules*. Дополнительные сведения можно получить в документации к пакету *module-init-tools*.

Если модуль загружен не вручную с помощью *insmod* или *modprobe*, а автоматически ядром, то в выводе *lsmod* к нему приписывается строка `autoclean`. Это говорит о том, что ядро удалит модуль, если он не будет использоваться в течение минуты.

Мы разобрали большой объем материала, и теперь у вас должны быть все инструменты, необходимые для компиляции и сопровождения своего собственного ядра.

III

Программирование

Инструментальные средства и практические приемы, описываемые в этой части книги, необязательно пригодятся всем пользователям, но любой желающий сможет овладеть ими и еще больше расширить возможности своей системы. Если раньше вам приходилось заниматься программированием, эта часть книги быстро поможет вам прийти в состояние готовности писать свои программы для Linux. Для тех, кто не занимался программированием, этот материал продемонстрирует некоторые выгоды, которые несет умение писать программы, и определенно доставит вам удовольствие. В главе 20 будут описаны два замечательных инструмента, предназначенных для работы с текстом, – редакторы *vi* и Emacs, которыми однозначно следует овладеть, даже если вы не собираетесь заниматься программированием. Кроме того, эта часть книги послужит ценным справочником, к которому можно обращаться по мере чтения других частей.

19

Текстовые редакторы

В этой главе мы познакомимся с некоторыми редакторами, используемыми для работы с текстовыми файлами. Из предыдущих глав вы уже знаете, что Linux просто изобилует разного рода файлами с настройками. Даже при том, что с каждым годом появляется все больше и больше инструментов настройки системы с графическим интерфейсом, тем не менее вам не удастся далеко уехать без владения навыками работы хотя бы в одном текстовом редакторе. Кроме того, если вам понадобится создавать текстовые документы с использованием настоящих языков форматирования, которые описываются в следующей главе, или писать свои собственные программы, как описано в главе 21, надобность в текстовых редакторах будет ощущаться еще острее. В этой главе будет рассмотрен не один текстовый редактор, и тому есть веские основания. Существуют большие и очень удобные редакторы, такие как XEmacs, но если нужно изменить в файле всего несколько символов, то затраты времени на запуск такого огромного зверя могут оказаться значительно больше, чем вы готовы подождать, и в этом случае лучше воспользоваться редактором меньшего размера, например *vi*. Может также случиться, что вы связаны с удаленной системой Linux посредством канала с небольшой пропускной способностью. В этом случае тоже может оказаться лучше пожертвовать удобством работы в редакторе в угоду более быстрой перерисовке, свойственной более простым редакторам.

Редактирование файлов с помощью vi

В этом разделе мы собираемся рассказать об использовании редактора *vi* (произносится как «ви-ай»). Редактор *vi* был первым настоящим экранным редактором для UNIX. Он прост, невелик по размеру и «вылизан». Если вы выполняете функции системного администратора, то знание *vi* необходимо: во многих аварийных ситуациях большие редакторы типа (X)Emacs недоступны (например, при загрузке Linux с аварийного диска).

Редактор *vi* основан на тех же принципах, что и многие другие приложения UNIX: каждая программа должна обеспечивать свою специфическую функцию и быть в состоянии взаимодействовать с другими программами. Например, в *vi* нет собственного средства проверки орфографии или заполнителя абзацев, но

эти функции реализованы в других программах, которые легко запустить из *vi*. Поэтому, несмотря на некоторую ограниченность, *vi* может взаимодействовать с другими программами и обеспечить практически любые необходимые функции.

На первый взгляд, *vi* несколько сложен и неуклюж. Однако его однобуквенные команды покажутся быстрыми и мощными, когда вы их изучите. В следующем разделе мы опишем Emacs – более гибкий редактор (фактически интегрированную рабочую среду), который можно освоить быстрее. Но помните, что умение работать с *vi* может оказаться решающим в ситуации, когда (X)Emacs окажется недоступным, поэтому мы настоятельно советуем изучить основы *vi*, какими бы непривычными они ни показались. Следует добавить, что в настоящее время есть ряд клонов *vi*, с которыми значительно удобнее работать, чем с первоначальным *vi*. Среди них наиболее популярен *vim*. Вероятнее всего, ваш дистрибутив устроен так, что при вызове *vi* вы фактически запускаете какой-то из этих новых редакторов. Мы будем говорить здесь только об основных функциях, которые применимы в любой версии *vi*. Освещение более новых версий можно найти в книге «Learning the vi Editor», написанной Линдой Лэм (Linda Lamb) и Арнольдом Роббинсом (Arnold Robbins) (O'Reilly).

Запуск vi

Давайте запустим *vi* и отредактируем файл. Синтаксис *vi* следующий:

```
vi filename
```

Например:

```
eggplant$ vi test
```

запустит *vi* для редактирования файла *test*. Ваш экран при этом должен выглядеть, как показано на рис. 19.1.

Колонка символов ~ показывает, что вы находитесь в конце файла.

***Рис. 19.1.** Новый файл, открытый в редакторе vi*

Вставка текста и перемещение по нему

При работе в *vi* вы всегда находитесь в одном из двух (или трех, в зависимости от того, как на это посмотреть) режимов, носящих названия: *командный режим* (*command mode*), *режим редактирования* (*edit mode*) и *расширенный командный режим*.

После запуска *vi* вы попадаете в командный режим. Как мы вскоре увидим, в этом режиме можно использовать ряд команд (обычно однобуквенных) для редактирования текста. Фактический ввод или модификация текста выполняются в ре-

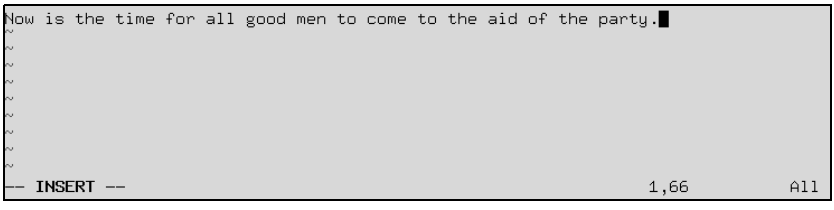


Рис. 19.2. *Ввод текста в редакторе vi*

жиме редактирования. Чтобы начать вставку текста, нажмите `i` (что переведет вас в режим редактирования) и введите текст (рис. 19.2).

Вставляя текст, можно вводить любое число строк (разумеется, нажимая после каждой строки клавишу Enter) и исправлять ошибки нажатием клавиши Backspace. Чтобы завершить редактирование и вернуться в командный режим, нужно нажать клавишу Esc.

В командном режиме можно пользоваться клавишами со стрелками для перемещения по документу. Допустимо также использование клавиш `h`, `j`, `k` и `l`, которые перемещают курсор влево, вниз, вверх и вправо соответственно.

Есть и другие способы вставки текста помимо команды `i`. Команда `a` («append») вставляет текст после текущей позиции курсора. Например, переместите курсор клавишей «стрелка влево» в положение между словами `good` и `men` (рис. 19.3).

Нажмите `a`, введите `wo` и вернитесь в командный режим, нажав Esc (рис. 19.4).

Чтобы открыть строку под текущей и начать ввод текста, используйте команду `o`. Нажмите `o` и введите еще одну-две строки (рис. 19.5).

Запомните, что вы всегда находитесь либо в командном режиме (в котором можно вводить такие команды, как `i`, `a` или `o`), либо в режиме редактирования (в котором вы вводите текст, а затем нажимаете клавишу Esc для возврата в команд-

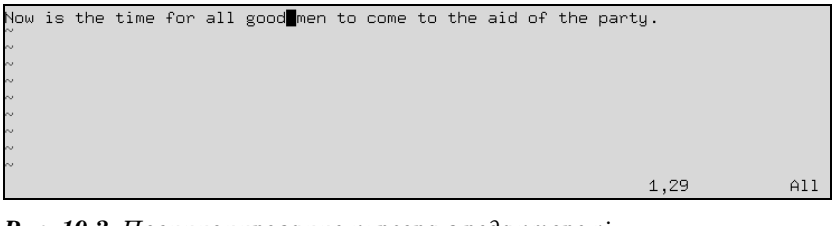


Рис. 19.3. *Позиционирование курсора в редакторе vi*

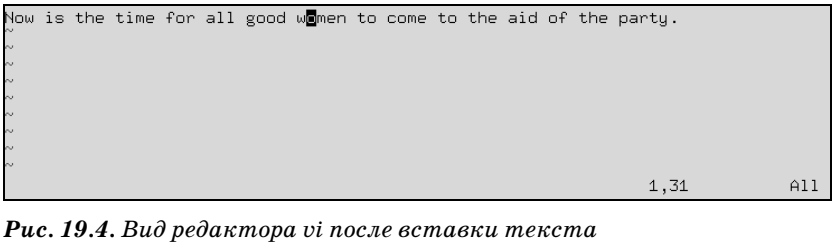


Рис. 19.4. *Вид редактора vi после вставки текста*

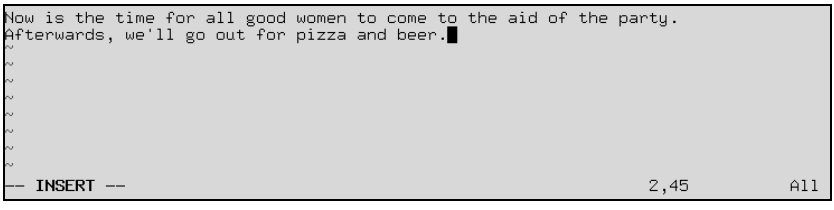


Рис. 19.5. *Вид редактора после добавления текста*

ный режим). Если вы забыли, в каком режиме находитесь, нажмите Esc. Это действие выведет вас из режима редактирования, если вы в нем находитесь, и не окажет никакого влияния, если вы были в командном режиме.

Удаление текста и отмена изменений

В командном режиме нажатие x удаляет символ под курсором. Если вы нажмете x пять раз, получится результат, продемонстрированный на рис. 19.6.

Теперь нажмите a и введите какой-нибудь текст, а затем нажмите клавишу Esc (рис. 19.7).

С помощью команды dd (то есть нажимая клавишу d дважды) можно удалять целые строки. Если курсор находится на второй строке, команда dd произведет результат, показанный на рис. 19.8.

Удаленный текст можно вставить заново командой p («put»). Если сейчас нажать p, то удаленная строка будет возвращена в буфер сразу за текущей. Если же ввести P (в верхнем регистре), то текст будет вставлен перед текущей строкой. По умолчанию команды p и P вставляют текст из буфера отмены, но можно копировать и вставлять текст из других буферов, как мы покажем позже.

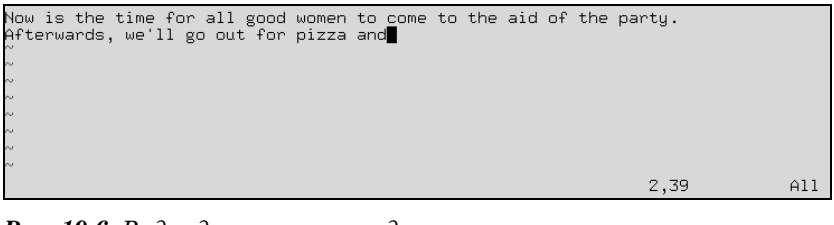


Рис. 19.6. *Вид редактора после удаления текста*

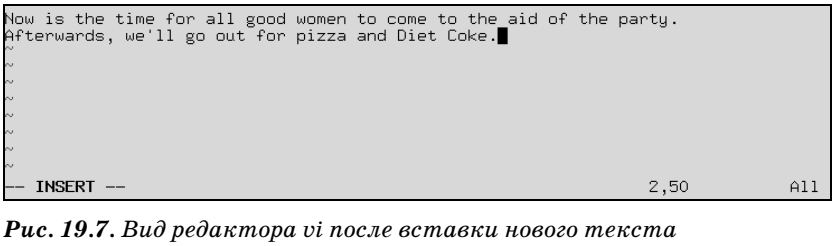


Рис. 19.7. *Вид редактора vi после вставки нового текста*

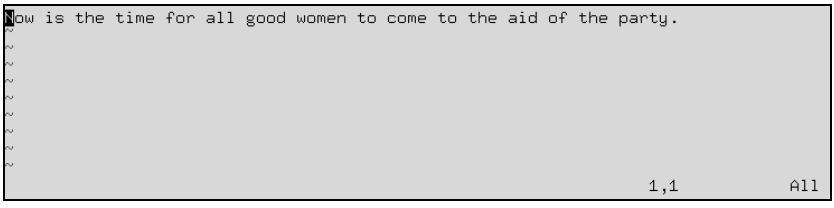


Рис. 19.8. *Вид редактора vi после удаления строки*

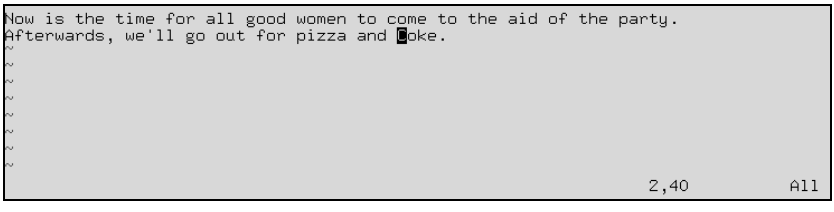


Рис. 19.9. *Вид редактора vi после удаления слова*

Команда `u` отменяет последнее произведенное изменение (в данном случае нажатие `u` после `dd` эквивалентно `p`). Если вы вставили большой кусок текста по команде `i`, то, нажав `u` сразу после возврата в командный режим, вы отмените вставку.

Для удаления слова под курсором используйте команду `dw`. Поместите курсор на слово `Diet` и введите `dw` (рис. 19.9).

Изменение текста

Заменить текст можно командой `R`, которая переписывает текст, начиная с положения курсора. Поместите курсор на первую букву в слове `pizza`, нажмите `R` и введите текст (рис. 19.10).

Команда `r` заменяет один символ под курсором. `r` не переводит редактор в режим редактирования, поэтому нет необходимости нажимать Esc для возврата в командный режим.

Команда `~` изменяет регистр буквы под курсором на противоположный. Если вы поместите курсор на первый символ `o` в слове `Now` в предыдущем примере и несколько раз нажмете `~`, результат будет выглядеть, как показано на рис. 19.11.

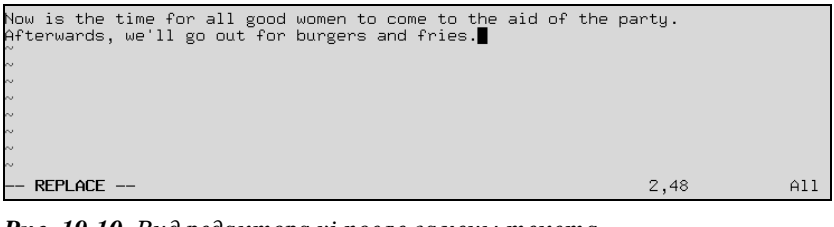


Рис. 19.10. *Вид редактора vi после замены текста*

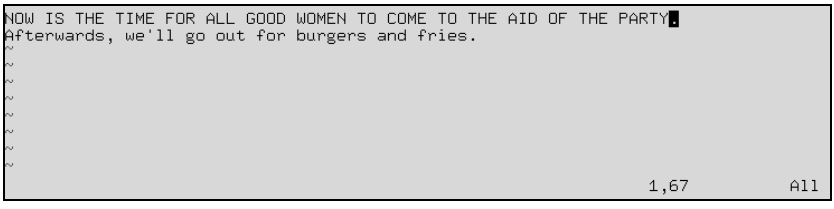


Рис. 19.11. *Вид редактора vi после изменения регистра символов*

Еще одна интересная команда – `cw`. Она изменяет слова, позволяя ввести новое слово и после нажатия Esc удалить остатки прежнего слова. Если новый текст длиннее, чем модифицируемый, пространство автоматически расширяется.

Команды перемещения по тексту

Вы уже знаете, как использовать клавиши со стрелками для перемещения по документу. Помимо этого команда `w` перемещает курсор к началу следующего слова; команда `b` перемещает его к началу текущего слова. Команда `0` (нуль) перемещает курсор к началу текущей строки, а команда `$` – к концу строки.

При редактировании больших файлов требуется перемещаться сразу на целый экран вперед или назад. Комбинация Ctrl+F перемещает курсор на один экран вперед, а Ctrl+B – на один экран назад.

Для перемещения в конец файла нажмите `G`. Можно также переместиться на произвольную строку: команда `10G` переместит курсор на 10-ю строку файла. Переместиться в начало файла можно командой `1G`.

Ввод символа `/`, а затем шаблона и нажатие Enter вызывает переход на первое совпадение текста с шаблоном за текущей позицией курсора. Например, поместив курсор на первую строку текста в нашем примере и введя `/burg`, мы переместим курсор в середину слова `burgers`. Если вместо `/` вводить `?`, то поиск осуществляется в обратном направлении.

Шаблон, следующий за командой `/` или `?`, является на самом деле регулярным выражением. Регулярные выражения являются мощным способом задания шаблонов для операций поиска и замены и используются во многих утилитах UNIX. Дополнительные сведения о регулярных выражениях можно найти ниже в разделе «Регулярные выражения». С помощью регулярного выражения можно, например, найти очередную заглавную букву, выполнив команду

```
/[A-Я]
```

Если то, что вы ищете, не является статической строкой, для задания искомого текста можно использовать регулярные выражения.

Команды перемещения можно соединять с другими командами, такими как удаление. Например, команда `d$` удалит текст от курсора до конца строки; `dG` удалит весь текст от курсора до конца файла.

Сохранение файлов и выход из vi

Большинство команд *vi*, работающих с файлами, вызываются из расширенного командного режима. Переход в этот режим осуществляется нажатием клавиши : (двоеточие) в командном режиме. В результате курсор перемещается на последнюю строку экрана и позволяет вводить различные расширенные команды.

Например, для записи редактируемого файла нужно ввести :w. Ввод : влечет переход в расширенный командный режим, а нажатие w и затем Enter завершает команду. Команда :wq записывает файл и осуществляет выход из *vi*. Команда ZZ (в командном режиме, без :) эквивалентна :wq, но проверяет сначала, изменился ли файл, и только тогда его записывает.

Чтобы выйти из *vi* без сохранения изменений, используется команда :q!. Просто команда :q позволяет завершить *vi*, но только если изменения в файле сохранены. Знак ! в команде :q! означает серьезность вашего намерения выйти из *vi*.

Переход к редактированию другого файла

Чтобы начать редактирование другого файла, используйте команду :e. Например, чтобы прекратить редактирование файла *test* и начать вместо него редактировать файл *foo*, выполните команду, как показано на рис. 19.12.

Если вы попытаетесь выполнить :e, не сохранив сначала файл, то получите сообщение об ошибке:

```
No write since last change (:edit! overrides)
```
(Изменения не сохранены (добавьте !, чтобы обойти проверку))

В таком случае можно выполнить :w, чтобы сохранить исходный файл, а затем выполнить :e, либо можно использовать команду :e! foo, которая заставит *vi* перейти к редактированию нового файла без сохранения изменений в текущем. Вторым способом удобно воспользоваться, когда в процессе редактирования файла вы вдруг почувствуете, что испортили его. Тогда выполнение команды :e! без указания имени файла приведет к отмене всех изменений и редактированию текущего файла с начала.

Вставка других файлов

С помощью команды :r можно включить в буфер *vi* содержимое другого файла. Например, команда

```
:r foo.txt
```

вставит содержимое файла *foo.txt* после текущей строки.

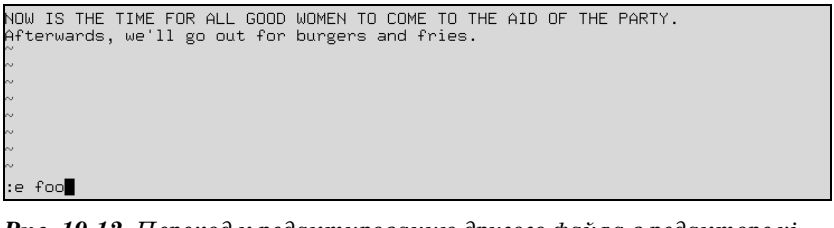


***Рис. 19.12.** Переход к редактированию другого файла в редакторе vi*

Выполнение команд оболочки

Команда `:!` позволяет ввести имя команды, выполняемой внутри *vi*. Например, команда

```
:! ls -F
```

выполнит команду `ls` и выведет результат на экран (рис. 19.13).

Команда `:r!` аналогична `:!`, но помещает данные, выводимые командой на стандартное устройство вывода, в буфер. Команда

```
:r! ls -F
```

дает результат, как показано на рис. 19.13.

Если нужно выполнить несколько команд оболочки, то зачастую проще использовать клавиши прерывания (обычно Ctrl+Z) при условии использования командной оболочки, поддерживающей управление заданиями, такой как *zsh* или *bash*.

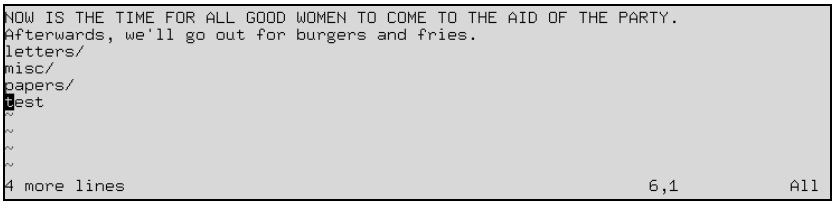


Рис. 19.13. *Вставка результатов работы команды в редактор vi*

Глобальный поиск и замена

В *vi* значительно больше функций, чем здесь описано; большинство этих функций реализованы путем сочетания простых функций, которые мы уже рассмотрели. Вот еще пара приемов, которые применяются многими пользователями *vi*.

Команда

```
:[x,y]s/шаблон_искомой_строки/строка_для_замены/флаги
```

ищет *шаблон_искомой_строки* в буфере между строками *x* и *y* и заменяет вхождения *шаблон_искомой_строки* текстом *строка_для_замены*. *шаблон_искомой_строки* является регулярным выражением; *строка_для_замены* – текстовый литерал, но может содержать некоторые специальные символы, указывающие на элементы исходного шаблона. Следующая команда заменит первое вхождение `weeble` на `wobble` в строках с 1-й по 10-ю включительно:

```
:1,10s/weeble/wobble
```

Вместо того чтобы указывать номера строк, можно использовать символ `%`, который является ссылкой на весь файл. Вместо `x` и `y` можно использовать и другие специальные символы. `$` указывает на последнюю строку файла. Если `x` или `y` не указаны, поиск и замена будут произведены в текущей строке.

В число используемых флагов *флаги* входят `g` для замены всех вхождений шаблона в каждой строке и `c` для запроса подтверждения каждой замены. В большин-

стве случаев вам понадобится флаг g, если только вашим намерением не является замена первого вхождения шаблона в каждой строке.

Для ссылки на строки можно также использовать *метки* (*marks*). Метки являются однобуквенными именами, присваиваемыми положениям курсора в документе. Если после установки курсора в некоторую позицию в файле ввести команду ma, в эту позицию будет установлена метка a. (Метки можно именовать любыми буквами a–z или A–Z.) Курсор можно переместить прямо на метку a командой `a (с обратным апострофом). Использование обычного апострофа ('a) приводит к перемещению курсора в начало строки, в которой находится метка a.

Метки позволяют «запоминать» положения курсора для обозначения области текста. Например, если вам нужно осуществить поиск и замену в участке текста, переместите курсор в начало текста, установите метку, переместите курсор в конец текста и выполните команду

```
:'a,.s/weeble/wobble
```

где 'a указывает на строку, содержащую метку a, а . ссылается на текущую строку.

Перемещение текста и использование регистров

Копировать и перемещать текст можно с помощью команд удаления (d или dd) и последующей команды вставки P, описанных выше. Например, если необходимо удалить 10 строк, начиная с той, где находится курсор, а затем вставить их в другое место, выполните команду 10dd (чтобы удалить 10 строк), переместите курсор в то место, где должен находиться текст, и введите p. Таким же образом можно и копировать текст: ввести сначала команду 10dd, а затем P (в том же положении курсора) – данная комбинация команд сначала удалит текст и затем сразу же его восстановит. После этого можно будет вставлять текст в любое место, перемещая курсор и всякий раз нажимая p.

Сходной с dd является команда yy, которая копирует («yanks») текст, не удаляя его. Для вставки скопированного текста используется команда p, так же как с dd. Но учтите, что каждая операция копирования удаляет из буфера ранее скопированный текст.

Команды удаления и копирования в буфер можно применять и к более общим областям, нежели строки. Напомним, что команда d удаляет текст с помощью команды перемещения; например, d$ удаляет текст от текущего положения курсора до конца строки. Аналогично y$ копирует текст от курсора до конца строки.

Допустим, вы хотите скопировать в буфер (или удалить) участок текста. Это можно сделать с помощью меток. Переместите курсор в начало копируемого текста и поставьте метку, например командой ma. Переведите курсор в конец копируемого текста и выполните команду y`a. В результате будет скопирован текст от позиции курсора до метки a. (Вспомните, что команда `a перемещает курсор на метку a.) Команда d, используемая вместо y, удалит текст от курсора до метки.

Удобнее всего вырезать, копировать и вставлять участки текста в *vi*, используя регистры. Регистр является именованным временным местом хранения текста, который необходимо скопировать в различные места, вырезая и вставляя участки текста в документе и т. д.

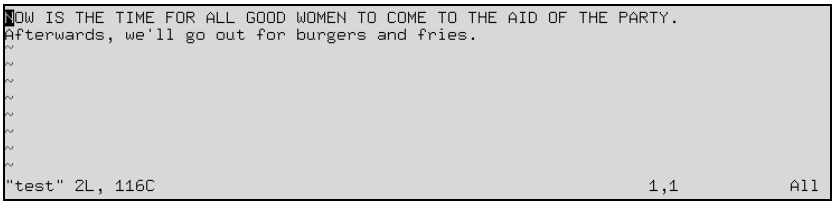


Рис. 19.14. *Вид редактора vi перед операцией копирования*

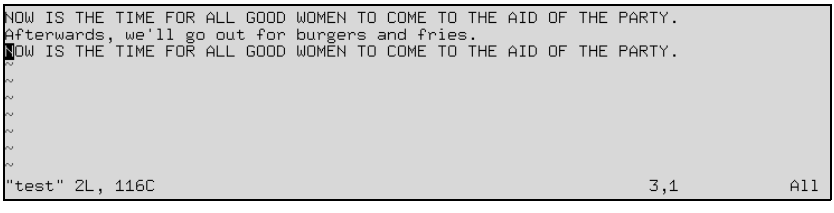


Рис. 19.15. *Вид редактора vi после операции копирования*

Имена регистров состоят из одной буквы в диапазоне a–z или A–Z. Команда " (двойные кавычки) задает обращение к регистру, затем следует имя регистра, например "a для регистра a. Большие и маленькие буквы относятся к одним и тем же регистрам, но при использовании маленьких букв прежнее содержимое регистра затирается, а при использовании больших букв происходит дописывание к регистру.

Например, если переместить курсор к первой строке нашего примера (рис. 19.14) и выполнить команду "ayy, текущая строка будет скопирована в регистр a. Затем можно переместить курсор во вторую строку и выполнить команду "ap, чтобы вставить текст из регистра a после текущей строки (рис. 19.15).

Аналогично команда "ay`a копирует текст от курсора до метки a в регистр a. Обратите внимание: между именами меток и регистров нет никакой связи!

Использование регистров позволяет копировать текст между файлами. Нужно просто скопировать текст в регистр, выполнить команду :e для перехода к редактированию нового файла и вставить текст из регистра.

Расширение возможностей vi

Есть много способов расширить возможности редактора *vi*. Большинство представленных нами команд можно обобщить на произвольные области текста. Как мы уже видели, такие команды, как d и y, действуют от позиции курсора до позиции, определяемой командой перемещения, например $ или G. (dG удаляет текст от курсора до конца файла.) Многие другие команды обрабатывают текст посредством команды перемещения таким же способом. Используя метки, можно обрабатывать любую область текста.

Как уже отмечалось ранее, *vi* является лишь текстовым редактором – в нем нет средств проверки орфографии, компиляции программ и других подобных функций. Однако *vi* запускает другие программы, которые могут служить расширением редактора. Команда

```
:x,y!command
```

выполняет указанную команду, для которой строки с *x* по *y* служат стандартным вводом, и заменяет строки стандартным выводом команды. Как и в команде `s` (поиск и замена), для задания номеров строк можно использовать специальные символы, такие как `%` и `$`.

Допустим, к примеру, вы хотите поставить символ цитирования (`>`) в начале всех строк некоторой области текста. Сделать это можно, написав короткий сценарий на языке командной оболочки или на языке Perl (см. раздел «Языки программирования и утилиты» главы 1), который считывает входные строки и выводит те же строки, предваряемые символом цитирования. (Можно также использовать команду *sed* – способов много.) Затем можно отправлять строки текста на этот фильтр, который будет заменять их цитированным текстом в *vi*. Если сценарий назвать *quote*, то нужно выполнить команду:

```
:'a,.!quote
```

которая пометит символами цитирования область текста между положением курсора и меткой `a`.

Ознакомьтесь с различными имеющимися командами расширенного режима. Команда `:set` позволяет устанавливать различные параметры. Например, `:set ai` включает автоматическое создание отступов в тексте (`:set noai` выключает его).

Можно задать команды расширенного режима (такие как `:set`), которые должны выполняться при запуске *vi*, поместив их в файл *.exrc* в домашнем каталоге. (Имя файла можно определить в переменной окружения `EXINIT`.) Например, файл *.exrc* может содержать строку

```
set ai
```

для включения автоматического создания отступов. Ставить в этом файле `:` перед командами расширенного режима не требуется.

Есть ряд хороших руководств и справочников по *vi*, как электронных, так и печатных. Дополнительные сведения можно поискать в «Learning the vi Editor». Одним из популярных сайтов, содержащих сведения о редакторе *vi*, является домашняя страница любителей *vi*: *http://thomer.com/vi/vi.html*. Домашняя страница редактора *vim* – *http://www.vim.org*.

Редактор (X)Emacs

Текстовые редакторы являются одними из самых важных приложений в мире UNIX. Они используются так часто, что многие проводят в редакторе больше времени, чем в какой-либо другой программе UNIX. То же справедливо для Linux.

Выбор редактора может стать сродни вопросу выбора религии. Есть много редакторов, но сообщество UNIX разделилось на две основные группы: лагерь приверженцев Emacs и лагерь *vi*. По причине не вполне интуитивного интерфейса *vi* многие (как новички, так и опытные пользователи) предпочитают Emacs. Однако те, кто давно работают в *vi* (и печатают одним пальцем), используют его эффективнее, чем более сложный редактор, каким является Emacs.

На одном краю спектра редакторов находится *vi*, а на другом – Emacs. Они значительно различаются по архитектуре и философии. Emacs отчасти является

плодом ума Ричарда Столмена (Richard Stallman), основателя Фонда свободного программного обеспечения и автора значительной доли программного обеспечения GNU.

Emacs – очень крупная система, в которой больше функций, чем в любом другом отдельном современном приложении UNIX (некоторые даже называют его не редактором, а интегрированной средой). В нем есть собственный интерпретатор языка LISP, который можно использовать для создания расширений редактора. (Многие внутренние функции Emacs написаны на Emacs LISP.) В Emacs есть расширения для чего угодно – от компиляции и отладки программ до работы с электронной почтой и поддержки X Window System и т. д. В Emacs есть также собственный электронный учебник и документация. Популярным введением в редактор Emacs служит книга Дебры Камерон (Debra Cameron), Билла Розенблатта (Bill Rosenblatt) и Эрика Рэймонда (Eric Raymond) «Learning GNU Emacs», O'Reilly.

В большинстве дистрибутивов Linux есть две версии Emacs. Первоначальной версией является GNU Emacs. Она разрабатывается до сих пор, но разработка, похоже, замедлилась. XEmacs больше по размеру, но значительно более дружествен к пользователю и лучше интегрирован с X Window System (несмотря на название, его можно запускать в командной строке). Если вы не очень стеснены в памяти и обладаете достаточно быстрым компьютером, то мы бы посоветовали пользоваться XEmacs. Еще одним достоинством XEmacs является то обстоятельство, что многие полезные пакеты, которые в случае GNU Emacs нужно загружать и устанавливать отдельно, уже входят в поставку XEmacs. Мы не будем говорить о различиях между ними, а остановимся на том, что присуще обоим редакторам. Упоминая в этом разделе *Emacs*, мы будем подразумевать любую из версий.

Запуск Emacs

Запуск GNU Emacs осуществляется просто:

```
$ emacs options
```

Аналогично XEmacs вызывается следующим образом:

```
$ xemacs options
```

В основном параметры вам не потребуются. Можно задать в командной строке имена файлов, но проще открыть их после запуска программы.

На жаргоне Emacs `C-x` означает Ctrl+X, а `M-p` эквивалентно Alt+P. Нетрудно догадаться, что `C-M-p` означает Ctrl+Alt+P.

Учитывая эти соглашения, нажмите `C-x`, а затем `C-f`, чтобы прочесть файл или создать новый. В результате нажатия клавиш в нижней части экрана будет выведено приглашение, показывающее текущий рабочий каталог. Теперь можно создать буфер, который в итоге станет новым файлом; назовем его *wibble.txt*, как показано на рис. 19.16.

В результате мы видим следующее: строка режима внизу указывает имя файла и тип буфера, в котором вы находитесь (в данном случае Fundamental). Emacs поддерживает много режимов редактирования. Fundamental является режимом по умолчанию для простых текстовых файлов, но есть режимы для редактирования исходных текстов на языке программирования C и TEX, для изменения ка-

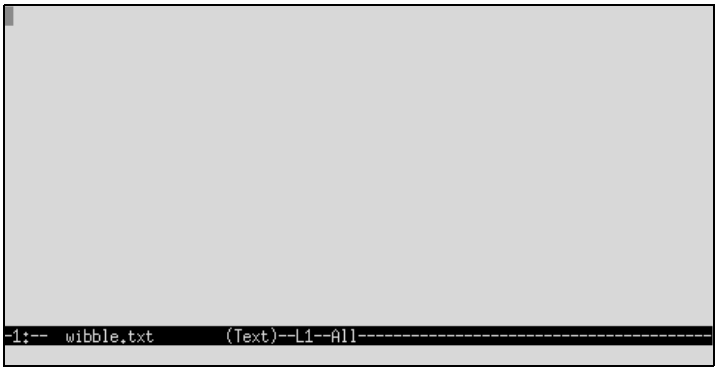


Рис. 19.16. *Вид редактора Emacs после открытия нового файла*

талогов и т. д. Как мы скоро увидим, для каждого режима существуют свои привязки клавиш и команды. Обычно Emacs определяет тип буфера, основываясь на расширении имени файла.

Справа от типа буфера вы видите слово `All`, означающее, что у вас перед глазами находится весь файл (который пуст). Обычно вы видите величину, показывающую ваше положение относительно начала файла в процентах.

Если вы запускаете Emacs в X Window System, то для редактора создается новое окно с меню вверху, полосами прокрутки и прочими удобствами. В разделе «Emacs и X Window System» далее в этой главе мы обсудим особенности работы Emacs в среде X.

Простые команды редактирования

Основные операции редактирования выполняются в Emacs проще, чем в *vi*. Клавиши со стрелками должны перемещать курсор по буферу. Если этого не происходит (когда Emacs не настроен для вашего терминала), используйте клавиши `C-p` (предыдущая строка), `C-n` (следующая строка), `C-f` (вперед на символ) и `C-b` (назад на символ).

Если вам неудобно использовать клавишу Alt, нажмите Esc, а затем `p`. Нажать и отпустить Esc эквивалентно удержанию клавиши Alt в нажатом положении.

В этом месте нужно сделать первое отступление в нашем путешествии по Emacs. Можно переназначить буквально каждую команду и клавишу в Emacs. При настройках по умолчанию команда `C-p` вызывает внутреннюю функцию *previous-line*, которая перемещает курсор (называемый также «point» – точка) к предыдущей строке. Однако нетрудно привязать к этим функциям другие клавиши или написать новые функции и привязать клавиши к ним и т. д. Если не сказано иное, то клавиши, о которых мы рассказываем, действуют в конфигурации Emacs по умолчанию. Ниже мы покажем, как переназначать клавиши для использования соответственно личным предпочтениям.

Вернемся к редактированию. Клавиши со стрелками или их эквиваленты перемещают курсор по текущему буферу. Начинайте печатать текст, и он будет вставляться в текущую позицию курсора. Нажатие клавиши Backspace или Delete

Рис. 19.17. *Содержимое буфера Emacs после ввода текста*

должно удалять текст в позиции курсора. Если этого не происходит, то исправить положение можно, опираясь на сведения в разделе «Настройка Emacs» далее в этой главе. Теперь введите текст, как показано на рис. 19.17.

Комбинации клавиш `C-a` и `C-e` перемещают курсор соответственно к началу и к концу текущей строки. `C-v` перемещает на страницу вперед, а `M-v` перемещает на страницу назад. Есть много других базовых команд редактирования, о которых можно узнать в электронной документации по Emacs (о ней мы вскоре скажем).

Выйти из Emacs можно с помощью команды `C-x C-c`. Это первая из встретившихся нам расширенных команд. Во многих командах Emacs требуется нажатие нескольких клавиш. Комбинация `C-x` служит «префиксом» для других клавиш. В данном случае нажатие `C-x` с последующим нажатием `C-c` приводит к окончанию работы Emacs, который предварительно спрашивает, действительно ли вы хотите выйти, не сохранив сделанные в буфере изменения.

Клавиши `C-x C-s` сохраняют текущий файл, а нажатие `C-x C-f` вызывает «поиск» нового файла для редактирования. Например, введя `C-x C-f`, можно получить приглашение:

```
Find file: /home/loomer/mdw/
```

в котором отображается текущий каталог. В ответ введите имя файла, который нужно открыть. Нажатие клавиши Tab вызывает функцию автодополнения имени файла, аналогичную той, что используется в командных оболочках *bash* и *zsh*. Например, после ввода

```
Find file: /home/loomer/mdw/.bash
```

и нажатия клавиши Tab откроется другой буфер, в котором будут перечислены все возможные завершения, как показано на рис. 19.18.

После ввода полного имени файла буфер `*Completions*` исчезает, и отображается новый редактируемый файл. Это пример того, как Emacs использует временные буферы для вывода информации. Если не предполагалось использовать текущий каталог, тогда, вместо того чтобы удалять всю строку, достаточно будет добавить еще один символ слэша к отображаемому пути и начать заново.

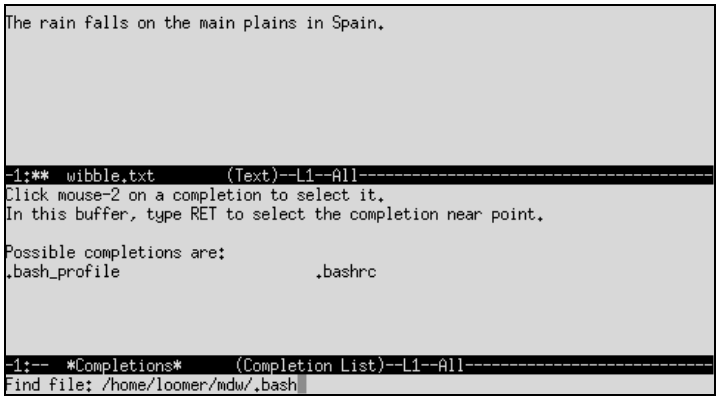


Рис. 19.18. *Буфер вариантов дополнения в Emacs*

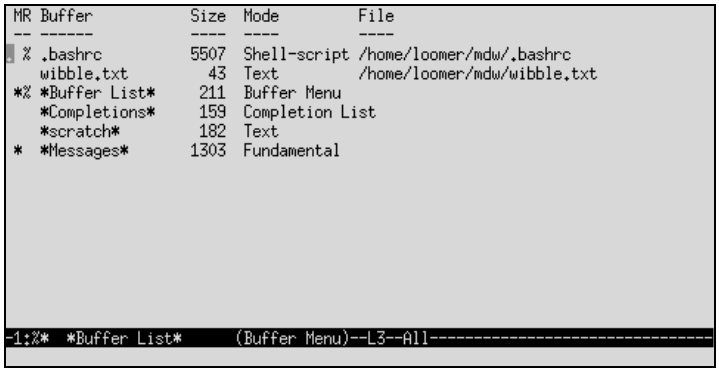


Рис. 19.19. *Список буферов в Emacs*

Emacs позволяет использовать при редактировании несколько буферов, в каждом из которых может содержаться свой файл. При загрузке файла командой `C-x C-f` для его редактирования создается новый буфер, при этом первоначальный буфер не удаляется.

Перейти в другой буфер можно командой `C-x b`, которая запрашивает имя буфера (обычно совпадающее с именем файла в буфере). Например, нажатие `C-x b` выводит приглашение:

```
Switch to buffer: (default wibble.txt)
```

Буфером по умолчанию является тот, в котором вы были до этого. Нажмите Enter для перехода в буфер по умолчанию или введите имя другого буфера. Команда `C-x C-b` выводит список буферов (в собственном буфере), как показано на рис. 19.19.

Появившееся меню буфера разделяет экран на два окна, переключаться между которыми можно командой `C-x o`. Одновременно можно открыть и более двух окон. Для того чтобы видеть одновременно только одно окно, перейдите в нужное окно и нажмите `C-x 1`. В результате остальные окна будут скрыты, но пере-

ключаться в них можно с помощью уже описанной команды C-x b. Нажатие C-x k приводит к удалению буфера из памяти Emacs.

Учебное руководство и оперативная подсказка

Если Emacs выглядит довольно сложным, то это из-за того, что он является очень гибкой системой. Прежде чем двигаться дальше, полезно познакомиться со встроенной оперативной подсказкой и учебным руководством по Emacs. Эта документация также доступна в виде книги «GNU Emacs Manual» Ричарда Столмана (Richard M. Stallman), GNU Press.

Команда C-h выводит в последней строке экрана список параметров подсказки. Повторное нажатие C-h выводит их описания. В частности, команда C-h t перемещает вас в учебное руководство по Emacs, которое написано ясно и расскажет о системе больше, чем мы сможем осветить в этой книге.

Изучив руководство по Emacs, вам следует освоить систему Info, в которой находится остальная документация по Emacs. Программа чтения Info вызывается командой C-h i. Гипотетическая страница Info может выглядеть так:

```
File: intercal.info, Node: Top, Next: Instructions, Up: (dir)

        This file documents the Intercal interpreter for Linux.

        * Menu:

        * Instructions:: How to read this manual.
        * Overview:: Preliminary information.
        * Examples:: Example Intercal programs and bugs.
        * Concept Index:: Index of concepts.
```

Как вы видите, помимо текста имеется меню для перехода к другим узлам. Нажав m и введя имя узла из меню, вы сможете прочесть текст соответствующего узла. Можно последовательно читать узлы, нажимая клавишу пробела, которая переводит к следующему узлу документа (указанному в информационной строке вверху буфера). В данном случае очередным узлом является первый узел меню – Instructions.

У каждого узла есть ссылка на родительский узел (Up), которым в данном случае является (dir), соответствующий каталогу страниц Info. Нажав u, вы переместитесь в родительский узел. Кроме того, у каждого узла есть ссылка на предыдущий узел, если он существует (в данном случае его нет). Команда p перемещает к предыдущему узлу. Команда l возвращает к последнему посещенному узлу.

Находясь в программе чтения Info, можно нажать ? для вывода списка команд и h для вывода краткого руководства по использованию системы. Поскольку вы запустили Info внутри Emacs, то можете использовать также и команды Emacs (например, C-x b для перехода в другой буфер).

Если система Info представляется вам загадочной и устаревшей, то учтите, что она разрабатывалась для работы на любых системах, включая те, в которых отсутствуют графические возможности и большие вычислительные мощности.

В Emacs есть и другая оперативная подсказка. Нажав C-h C-h, вы получите список параметров подсказки. Одной из них является C-h k, после выбора которой можно нажать клавишу и получить документацию по связанной с ней функции.

Удаление, копирование и перемещение текста

Перемещать и дублировать блоки текста в Emacs можно различными способами. В этих методах используется объект *«метка»* (*mark*), представляющий собой сохраненное в памяти положение курсора. Метки можно устанавливать с помощью различных команд. Участок текста между текущим положением курсора (point) и меткой называется *областью* (*region*).

Метку можно установить клавишами C-@ (или C-пробел в большинстве систем). Перемещение курсора в некоторое положение и нажатие C-@ устанавливает метку в этом положении. Теперь можно переместить курсор в другое место документа, и область будет определена как текст между меткой и текущим положением курсора.

Действие многих команд Emacs распространяется на область. Наиболее важные из них относятся к удалению и копированию текста. Команда C-w удаляет текущую область и помещает ее в *кольцо удаления* (*kill ring*). Кольцо удаления является списком удаленных текстовых блоков. Текст можно затем *вставить* (*yank*) в другое место командой C-y. (Семантика термина «yank» различна в *vi* и Emacs. В *vi* «yank» означает добавление участка текста к регистру отмены удаления без изъятия его из редактируемого текста, в то время как в Emacs «yank» означает вставку (paste) текста.) Используя кольцо удаления, можно вставлять не только последний удаленный блок, но и предшествующие удаленные блоки.

Введите, например, в буфер Emacs текст, как показано на рис. 19.20.

Теперь переместите курсор в начало второй строки («*Here is a line...*») и установите метку командой C-@. Переместите курсор в конец строки (командой C-e) и удалите выделенную область текста командой C-w, как показано на рис. 19.21.

Чтобы вставить только что удаленный текст, переместите курсор в конец буфера и нажмите C-y. Строка должна быть вставлена на новое место, как показано на рис. 19.22.

Повторное нажатие C-y приводит к многократной вставке текста.

Копирование текста можно производить аналогичным образом. Используя M-w вместо C-w, можно копировать область текста в кольцо удаления без уничтоже-

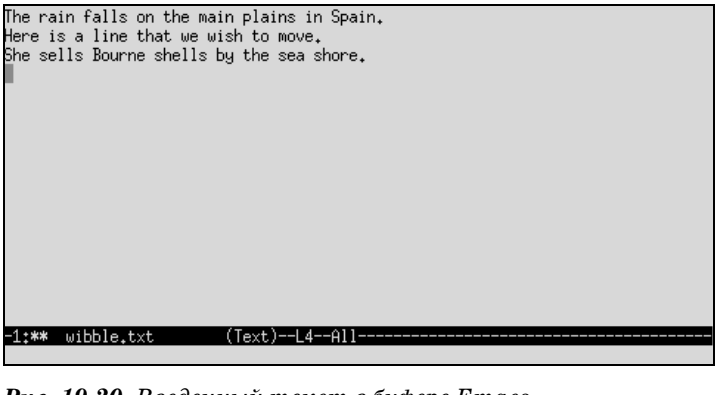


Рис. 19.20. *Введенный текст в буфере Emacs*

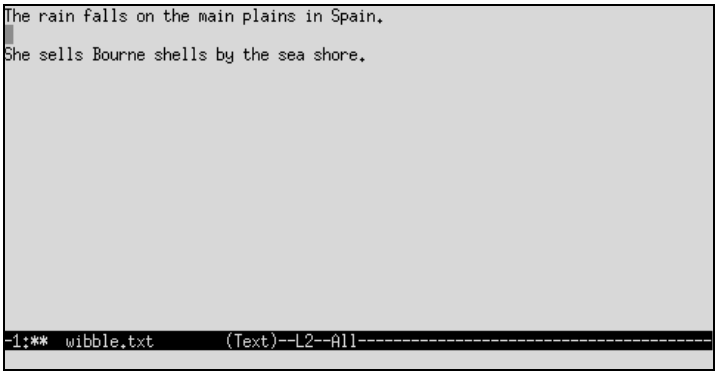


Рис. 19.21. *Буфер Emacs после операции удаления*

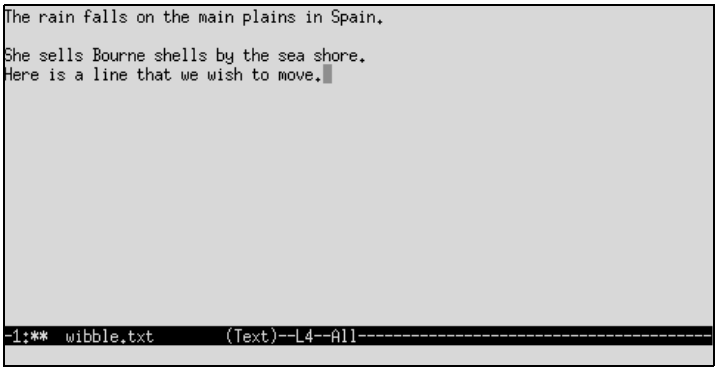


Рис. 19.22. *Буфер Emacs после операции вставки*

ния ее в буфере. (Напомним, `M-w` означает, что прежде чем нажать `w`, нужно нажать и удерживать клавишу Alt или нажать и отпустить Esc.)

Текст, удаляемый с помощью других команд, например `C-k`, также помещается в кольцо удаления. Это значит, что для перемещения участка текста не обязательно устанавливать метку и использовать `C-w`, можно использовать любую команду удаления.

Для вставки ранее удаленных участков текста (сохраненных в кольце удаления) выполните команду `M-y` после вставки командой `C-y`. Команда `M-y` заменяет вставленный текст предыдущим блоком из кольца удаления. При повторном нажатии `M-y` происходит циклический перебор содержимого кольца удаления. Эта функция полезна при перемещении или копировании нескольких блоков текста.

В Emacs есть также более общий механизм регистров, аналогичный имеющемуся в *vi*. Помимо прочего, эту функцию можно использовать для сохранения текста, который вы собираетесь вставить позже. Имя регистра состоит из одного символа. В следующем примере используется имя `a`:

1. Установите метку в начало сохраняемого текста, нажав одновременно клавиши Ctrl и «пробел» (или, если это не работает, нажмите `C-@`).

2. Переместите курсор в конец сохраняемой области.
3. Нажмите C-x x, а затем введите имя регистра (в данном случае a).
4. Если вы хотите вставить текст в каком-либо месте, нажмите C-x g, а затем введите имя регистра a.

Поиск и замена

Чаще всего поиск строки осуществляется в Emacs командой C-s. Она инициирует так называемый инкрементный поиск. Вы начинаете вводить символы, которые хотите найти, и при вводе каждого символа Emacs осуществляет поиск в прямом направлении строки, состоящей из всех введенных вами символов. При возникновении ошибки нажмите клавишу Delete и продолжайте ввод правильных символов. Если строка не найдена, Emacs подает звуковой сигнал. Если вхождение найдено, но вы хотите продолжить поиск для другой строки, снова нажмите C-s.

Таким же образом можно осуществлять поиск и в обратном направлении, используя команду C-r. Есть несколько других типов поиска, в том числе поиск регулярных выражений, который можно инициировать, нажав M-C-s. Он позволяет отыскивать выражения типа jo.*n, которое соответствует, например, именам John, Joan и Johann (по умолчанию при поиске не различаются верхний и нижний регистры символов).

Для замены строки введите команду M-%. Вам будет показана та строка, которая находится в буфере в данный момент, и предложено ввести строку, которой ее нужно заменить. Emacs показывает все места в буфере, в которых обнаруживается заменяемая строка, запрашивая подтверждение на ее замену. Нажмите «пробел» для замены строки, клавишу Delete – для пропуска и точку – для прекращения поиска.

Если вы хотите заменить все вхождения строки, следующие за текущим положением в буфере, без запроса подтверждений, введите M-x replace-string. (Комбинация M-x позволяет ввести имя функции Emacs и выполнить ее, не используя привязку к клавишам. Многие функции Emacs доступны только через M-x, если вы сами не привяжете их к клавишам.) Замену с поиском по регулярному выражению можно выполнить командой M-x replace-regexp.

Макросы

Название Emacs частично происходит от слова *macros*. Макросы являются простой, но мощной возможностью, которая превращает работу в Emacs в удовольствие. Если вы собираетесь часто осуществлять какие-либо повторяющиеся операции, нажмите C-x (, затем выполните необходимые действия и нажмите C-x). Две команды C-x с открывающей и закрывающей скобками запоминают все ваши нажатия клавиш. Повторно выполнить команды можно, введя C-x e.

Приведем простой пример, который можно применить к любому текстовому файлу. В этом примере первое слово каждой строки записывается с прописной буквы:

1. Нажмите C-x (, чтобы начать запись макроса.
2. Нажмите C-a, чтобы переместить курсор в начало текущей строки. При каждом выполнении макроса нужно быть уверенным, что курсор находится в нужном месте. Нажимая C-a, вы можете быть уверены, что макрос начнет выполняться в начале строки, то есть там, где нужно.

3. Нажмите M-c, чтобы сделать прописной первую букву первого слова.
4. Снова нажмите C-a, чтобы вернуться к началу строки, и C-n или «стрелка вниз», чтобы перейти к началу следующей строки. Таким образом, в следующий раз исполнение макроса гарантированно начнется с нужного места.
5. Нажмите C-x), чтобы закончить запись макроса.
6. Нажмите несколько раз C-x e, чтобы последующие строки начинались с прописной буквы. Либо несколько раз нажмите C-u, а затем C-x e. Повторное нажатие C-u является префиксом, вызывающим многократное повторение следующей команды. Если вы дойдете до конца документа, а макрос все еще будет выполняться, ничего страшного не произойдет, Emacs подаст звуковой сигнал и прекратит выполнение макроса.

Выполнение команд и программирование в Emacs

Emacs предоставляет интерфейсы ко многим программам, которые можно запускать в буфере Emacs. Например, есть режимы работы Emacs для чтения и отправки электронной почты, чтения телеконференций, компиляции программ и взаимодействия с командной оболочкой. С некоторыми из этих функций мы познакомимся в данном разделе.

Для отправки из Emacs электронной почты нажмите C-x m. В результате будет открыт буфер, который позволит составить и отправить почтовое сообщение, как показано на рис. 19.23.

Просто введите в этот буфер ваше сообщение и отправьте его командой C-c C-s. Можно вставлять текст из других буферов, расширять интерфейс с помощью собственных функций Emacs LISP и т. д. Более того, в Emacs есть режим RMAIL, позволяющий читать электронную почту прямо в редакторе, но мы не станем рассказывать о нем, поскольку большинство пользователей предпочитают отдельные почтовые программы. (Обычно эти программы предлагают выбрать Emacs в качестве редактора почтовых сообщений.)

С почтовым интерфейсом RMAIL сходен интерфейс GNUS – программы чтения телеконференций, базирующейся на Emacs и запускаемой командой M-x gnus. После запуска (и некоторой обработки файла *.newsrc*) появляется список телекон-

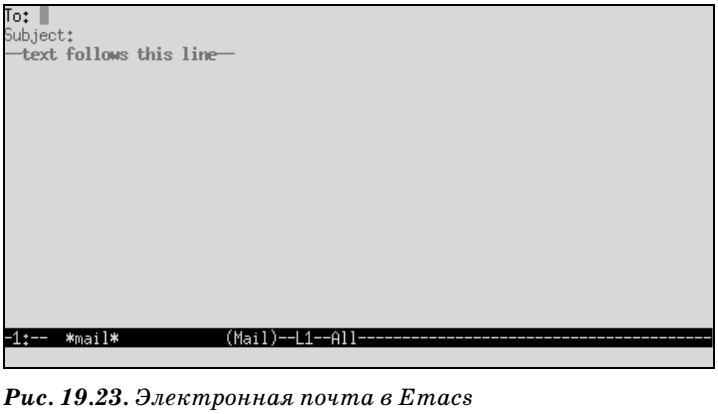


Рис. 19.23. *Электронная почта в Emacs*

Рис. 19.24. *Чтение телеконференций в Emacs*

ференций вместе со счетчиком непрочитанных статей для каждой из них, как показано на рис. 19.24.

GNUS служит примером мощи применения интерфейсов Emacs к другим инструментам. В вашем распоряжении оказываются возможности навигации, поиска и создания макросов, предоставляемые Emacs, наряду со специфическими комбинациями клавиш, соответствующими используемому вами инструменту.

Используя клавиши со стрелками, можно выбрать телеконференцию для чтения. Нажмите клавишу «пробел», чтобы начать чтение статей выбранной телеконференции. Будет выведено два буфера, в одном из которых отображается список статей, а в другом – текущая статья.

Команды `n` и `p` позволяют переместиться к следующей и предыдущей статьям, команды `f` и `F` служат для отправки ответа на текущую статью в телеконференцию (не включая или включая текст оригинала, соответственно), а `r` и `R` служат для ответа на статью по электронной почте. В GNUS есть много других команд, список которых можно получить командой `C-h m`. Если вы привыкли пользоваться такой программой чтения телеконференций, как `rn`, то GNUS не покажется вам непривычной.

В Emacs есть ряд режимов для редактирования файлов различных типов. Например, есть режим C для редактирования исходных текстов на языке программирования C и режим TEX для редактирования (сюрприз!) исходного кода TEX. В каждом из этих режимов есть функции, облегчающие редактирование файлов соответствующего типа.

Например, в режиме C можно выполнить команду `M-x compile`, которая по умолчанию запускает `make -k` в текущем каталоге и перенаправляет сообщения об ошибках в другой буфер. Например, в буфере компиляции может содержаться следующий текст:

```
cd /home/loomer/mdw/pgmseq/
        make -k
        gcc -O -O2 -I. -I../include -c stream_load.c -o stream_load.o
        stream_load.c:217: syntax error before `struct'
        stream_load.c:217: parse error before `struct'
```

Можно переместить курсор на строку, содержащую сообщение об ошибке, нажать `C-c C-c`, и курсор переместится на выбранную строку в соответствующем буфере с исходным текстом (при необходимости откроется буфер с соответствующим файлом исходного текста). Теперь можно редактировать и компилировать программы, не выходя из Emacs.

Emacs обеспечивает также полный интерфейс к отладчику `gdb`, который описан в разделе «Использование Emacs с gdb» главы 21.

Обычно Emacs выбирает для буфера режим, основываясь на расширении имени файла. Например, редактирование файла с расширением *.c* автоматически устанавливает для буфера режим C.

Режим Shell является одним из наиболее популярных расширений Emacs. Он позволяет взаимодействовать с командной оболочкой в буфере Emacs, используя команду `M-x shell`. Редактирование командных строк, удаление и вставку можно осуществлять стандартными командами Emacs. Одиночные команды оболочки можно выполнять из Emacs, используя `M-!`. Если вместо этой команды использовать `M-|`, то заданной команде в качестве стандартного ввода передается по каналу содержимое текущей области. Это общий интерфейс запуска подпрограмм из Emacs.

Настройка Emacs

Электронной документации Emacs должно быть достаточно, чтобы больше узнать об этой системе и освоиться с работой в ней. Однако иногда трудно найти самые полезные советы, чтобы начать работать. Ниже мы представим краткую сводку некоторых настроек, которые облегчают жизнь многим пользователям Emacs.

Личные настройки Emacs находятся в файле *.emacs*, расположенном в домашнем каталоге пользователя. Этот файл должен содержать программный код на языке Emacs LISP, который выполняет или определяет функции, настраивающие среду Emacs. (Если вы никогда не писали на LISP, не беспокойтесь: большинство настроек, в которых он используется, очень просты.)

Чаще всего пользователи изменяют привязку клавиш. Например, если вы используете Emacs для редактирования документов SGML, вы можете привязать клавиши `C-c s` для переключения в режим SGML. Поместите в свой файл *.emacs* следующие строки:

```
; C-c и s переведут буфер в режим SGML.
        (global-set-key "\C-cs" 'sgml-mode)
```

Комментарии в Emacs LISP начинаются с символа «точка с запятой». После комментария следует команда глобальной установки клавиш (*global-set-key*). Теперь вам не понадобится вводить длинную последовательность `M-x sgml-mode`, чтобы начать редактирование SGML. Нужно лишь нажать клавиши `C-c s`. Они сработают в любом месте Emacs независимо от режима текущего буфера, поскольку установлены как глобальные. (Конечно, Emacs может распознать файл SGML или XML по расширению и автоматически перевести его в режим SGML.)

Другая настройка, которая вам может понравиться, делает текстовый режим режимом по умолчанию и включает дополнительный режим autofill, который заставляет текст, слишком длинный для одной строки, автоматически переноситься на следующую строку:

```
; Make text mode the default, with auto-fill
        (setq default-major-mode 'text-mode)
        (add-hook 'text-mode-hook 'turn-on-auto-fill)
```

Возможно, не всегда желательно глобальное назначение клавиш. При работе в режимах TEX, C и других вы обнаружите, что есть полезные вещи, которые

нужны только в одном режиме. Сейчас мы определим простую LISP-функцию, вставляющую некоторые символы в исходный программный код на языке C, а затем для удобства привяжем эту функцию к клавише:

```
(defun start-if-block()
        (interactive)
        (insert "if () {\n}\n")
        (backward-char 6)
)
```

Мы начали с того, что определили функцию как интерактивную, чтобы ее мог вызывать пользователь, иначе ее можно было бы использовать только посредством других функций. Затем с помощью функции `insert` мы помещаем в буфер C следующие символы:

```
if ( ) {
       }
```

Строки в Emacs могут содержать стандартные управляющие символы C. Здесь мы использовали `\n` для перехода к новой строке.

Теперь у нас есть шаблон для блока `if`. Для красоты наша функция перемещает курсор на шесть символов назад, помещая его внутрь скобок, чтобы сразу можно было начать ввод выражения.

Нашей задачей было облегчить ввод этих символов, поэтому привяжем функцию к клавише:

```
(define-key c-mode-map "\C-ci" 'start-if-block)
```

Функция `define-key` привязывает клавишу к функции. Задав `c-mode-map`, мы определили, что клавиша действует только в режиме C. Существуют также привязки клавиш `tex-mode-map` для режима TEX, `lisp-mode-map`, которая вас заинтересует, если вы намерены серьезно заняться своим файлом *.emacs*, и т. д.

Если вы хотите писать собственные функции на Emacs LISP, то следует прочесть страницы Info для *elisp*, имеющиеся в вашей системе. Есть две хорошие книги о том, как писать функции на Emacs LISP: «An Introduction to Programming in Emacs Lisp» Роберта Чассела (Robert J. Chassel), GNU Press, и «Writing GNU Emacs Extensions» Боба Гликстайна (Bob Glickstein), O'Reilly.

Теперь рассмотрим важную настройку, которая может вам понадобиться. На многих терминалах клавиша Backspace посылает символ `C-h`, который является клавишей подсказки в Emacs. Чтобы исправить это, следует изменить внутреннюю таблицу, используемую Emacs для интерпретации клавиш:

```
(keyboard-translate ?\C-h ?\C-?)
```

Довольно таинственный код. `\C-h` узнаваем как клавиша Ctrl, нажатая вместе с `h`, которая производит тот же код ASCII (8), что и клавиша Backspace. `\C-?` представляет клавишу Delete (код ASCII 127). Не путайте этот вопросительный знак с вопросительными знаками перед символом обратного слэша. `?\C-h` означает «код ASCII, соответствующий `\C-h`». С тем же результатом можно непосредственно задать `8`.

Теперь удаление будет происходить и при нажатии Backspace, и при нажатии `C-h`. Вы потеряли клавишу вызова подсказки. Поэтому другой полезной настройкой

была бы привязка к C-h другой клавиши. Можно использовать C-\, которая редко для чего-либо используется. Нужно удваивать обратную косую черту, задавая ее в качестве клавиши:

```
(keyboard-translate ?\C-\\ ?\C-h)
```

В системе X Window есть способ изменить код, посылаемый клавишей Backspace, с помощью команды xmodmap, но мы оставим это для самостоятельных исследований. Это не вполне переносимое решение (поэтому мы не можем показать вам гарантированно работающий пример), и, на ваш взгляд, оно может оказаться слишком сильнодействующим (при этом изменяется значение клавиши Backspace в оболочке xterm и во всех других местах).

Вы можете применять и другие привязки клавиш. Например, вы захотите использовать клавиши C-f и C-b для прокрутки страницы вперед и назад, как в *vi*. Для этого в файл *.emacs* можно включить следующие строки:

```
(global-set-key "\C-f" 'scroll-up)
        (global-set-key "\C-b" 'scroll-down)
```

Мы снова должны предупредить: старайтесь не переопределять клавиши, которые имеют другое важное применение. (Один из способов узнать, что делает клавиша в текущем режиме, – выполнить команду C-h k. Следует также учесть, что клавиши могут быть определены и в других режимах.) В частности, вы потеряете доступ ко многим функциям, если измените привязку префиксных клавиш, начинающих команду, таких как C-x и C-c.

Если вы действительно хотите расширить некоторый режим множеством новых команд, можно создать собственные префиксные клавиши. Например, можно использовать:

```
(global-unset-key "\C-d")
        (global-set-key "\C-d\C-f" 'my-function)
```

Прежде всего, нужно «отвязать» клавишу C-d (которая удаляет символ под курсором), чтобы использовать ее в качестве префикса для других клавиш. Теперь нажатие C-d C-f будет выполнять my-function.

Возможно, вы захотите использовать для редактирования «обычных» файлов другие режимы, помимо Fundamental и Text. Например, режим Indented Text осуществляет автоматический отступ строк текста по отношению к предыдущей строке (как в функции *vi* :set ai). Чтобы этот режим включался по умолчанию, введите такие строки:

```
; Default mode for editing text
        (setq default-major-mode 'indented-text-mode)
```

Следует также переопределить клавишу Enter, чтобы новая строка текста начиналась с отступа:

```
(define-key indented-text-mode-map "\C-m" 'newline-and-indent)
```

Emacs также поддерживает дополнительные режимы, которые используются вместе с основными. Например, режим Overwrite является дополнительным и обеспечивает замену текста в буфере вместо вставки. Чтобы привязать клавишу C-r к переключению режима вставки-замены, используйте команду

```
; Toggle overwrite mode.
        (global-set-key "\C-r" 'overwrite-mode)
```

Другим дополнительным режимом является Autofill, который автоматически переносит строки при вводе. Это означает, что вместо нажатия Enter в конце каждой строки можно продолжать ввод текста, а Emacs будет автоматически разбивать текст на строки. Для включения режима Autofill используйте команды:

```
(setq text-mode-hook 'turn-on-auto-fill)
        (setq fill-column 72)
```

Эти команды включают режим Autofill при входе в режим Text (с помощью функции *text-mode-hook*). Они также устанавливают длину строки в 72 символа.

Регулярные выражения

Даже небольшое число приемов, использующих регулярные выражения, может значительно повысить эффективность поиска в тексте и выполнения групповых замен. Долгое время регулярные выражения ассоциировались только с утилитами и языками программирования в UNIX, но теперь они появляются и в других средах, например в Microsoft .NET, но только UNIX дает возможность пользоваться ими в самых разнообразных местах, например в текстовых редакторах и команде `grep`, где их могут применять обычные пользователи.

Допустим, вы просматриваете файл, содержащий сообщения электронной почты. Вы подписались на целый ряд почтовых списков рассылки с такими названиями, как `gyro-news` и `gyro-talk`, поэтому ищете строки `Subject`, в которых есть последовательность символов `gyro-`. Можно воспользоваться текстовым редактором или командой `grep` для поиска выражения

```
^Subject:.*gyro-
```

Это означает: «искать строки, начинающиеся с `Subject:`, после чего следует любое количество любых символов, а затем `gyro-`». Регулярное выражение состоит из нескольких частей, одни из которых воспроизводят обычный текст, а другие передают общие понятия, такие как «начало строки». На рис. 19.25 показано, что означают эти части и как они действуют вместе.

Чтобы дать представление о том, насколько мощными и сложными могут быть регулярные выражения, разовьем этот пример для осуществления более узкого поиска. Допустим, нам известно, что почтовые списки по гироскопам рассылают сообщения, строка `Subject` которых начинается с названия почтового списка, за-

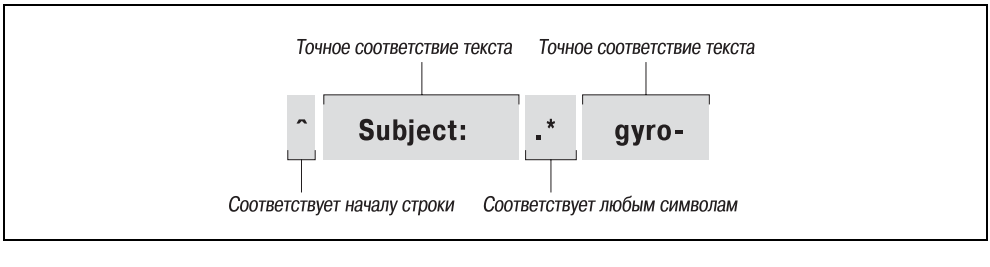


Рис. 19.25. *Простое регулярное выражение*

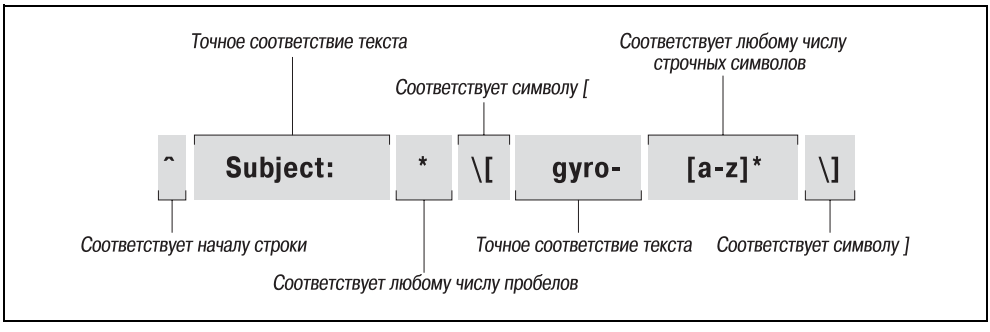


***Рис. 19.26.** Регулярное выражение с большим количеством частей*

ключенного в квадратные скобки, например `Subject: [gyro-news]` или `Subject: [gyro-talk]`. Мы можем вести поиск именно таких строк:

```
^Subject: *\[gyro-[a-z]*\]
```

На рис. 19.26 показано, что означают части, из которых состоит это регулярное выражение. Мы отметим здесь несколько интересных моментов.

В регулярных выражениях скобки, крышки и звездочки представляют собой специальные символы. Квадратные скобки используются для обозначения целых классов символов, которые нужно искать. Например, `[a-z]` представляет «любую букву в нижнем регистре». Нам не нужно, чтобы скобка перед `gyro` выражала этот особый смысл, поэтому мы помещаем перед ней символ обратного слэша; это называется *экранированием* (*escaping*). (Иными словами, мы позволяем квадратной скобке избежать рассмотрения в качестве метасимвола регулярного выражения.)

Первая звездочка в нашем выражении следует за пробелом, что означает «соответствует любому количеству смежных пробелов». Вторая звездочка следует за классом `[a-z]`, поэтому она применяется ко всей этой конструкции. Сама по себе конструкция `[a-z]` соответствует одному и только одному символу нижнего регистра. Вместе со звездочкой `[a-z]*` означает «соответствие любому количеству символов нижнего регистра, идущих подряд».

Чтобы научиться изощренно применять регулярные выражения, нужны недели. Рекомендуем прочесть книгу Джеффри Фридла (Jeffrey Friedl) «Mastering Regular Expressions».[1]

Emacs и X Window System

До сих пор разговор шел о свойствах Emacs, которые доступны как в текстовой консоли, так и в графической среде X Window. Теперь настало время рассмотреть свойства, которыми обладает Emacs для поддержки X Window.

[1] Дж. Фридл «Регулярные выражения», 3-е издание. – Пер. с англ. – СПб.: Символ-Плюс, 2008.

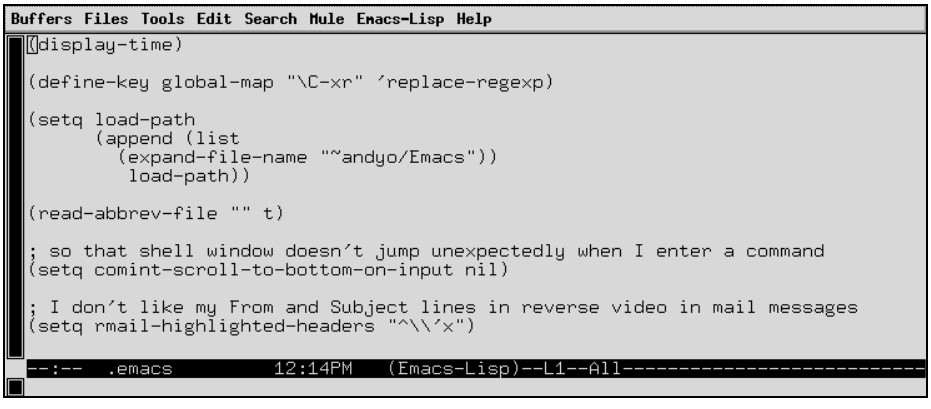


Рис. 19.27. *Окно Emacs*

Функций X в Emacs становится все больше и больше. Теперь они включают раскрывающиеся меню, различные шрифты для разных частей окна и полную интеграцию функций работы с выделенным текстом среды X. Большинство понятий, которые здесь рассматриваются, главным образом относятся к редактору Emacs, так как редактор XEmacs включает в себя встроенные меню для доступа ко многому из того, что здесь описывается. Тем не менее множество параметров настройки, описываемых в этом разделе, в равной мере относятся к обеим версиям Emacs.

Начнем с того, что определим цветовую гамму для разных частей окна Emacs. Попробуйте выполнить команду:

```
eggplant$ emacs -bg ivory -fg slateblue -ms orangered -cr brown
```

Тем самым вы установили цвет фона, цвет текста, цвет указателя мыши и цвет курсора, соответственно. Курсор – это маленький прямоугольник, появляющийся в окне, который представляет собой так называемую точку (point) в Emacs, то есть место, куда вводится набираемый текст. К цветам мы еще вернемся.

При запуске окно Emacs содержит строку меню вверху и полосу прокрутки с правой стороны (рис. 19.27).

Полоса прокрутки работает так же, как полоса прокрутки в `xterm`. Строка меню предлагает обычные функции. В некоторых режимах редактирования, таких как C и TEX, есть свои собственные раскрывающиеся меню. Меню не документированы, поэтому для выяснения особенностей их работы вам придется поэкспериментировать.

Меню приходит на помощь, когда вы хотите воспользоваться функцией, для которой нет простой комбинации клавиш, или когда вы забыли эту комбинацию. Например, если вы редко прибегаете к поиску с помощью регулярных выражений (очень мощная возможность, которая стоит времени, потраченного на ее изучение), самый простой способ вызвать эту функцию – это выбрать пункт Regexp Search в меню Edit.

Другой полезный элемент – пункт Choose Next Paste в меню Edit (в некоторых версиях это пункт Select and Paste), позволяющий получить то, что недоступно другими

способами: список всех фрагментов текста, которые вы недавно вырезали. Иными словами, вам показывается кольцо удалений. Вы можете выбрать текст, который хотите вставить, и при следующем нажатии C-y он будет помещен в буфер.

Если вы устали от полосы прокрутки и меню, попрощайтесь с ними и поместите следующий программный код на языке LISP в ваш файл *.emacs*:

```
(if (getenv "DISPLAY")
        (progn (menu-bar-mode -1)
        (scroll-bar-mode -1))
        )
```

Следующие интересные возможности X относятся к мыши. Вы можете вырезать и вставлять текст так же, как и в xterm. Можно обмениваться текстом между различными окнами; если вам надо поместить результаты какой-либо команды xterm в файл, вы можете скопировать их из xterm и вставить в буфер Emacs. Кроме того, с любым скопированным обычным образом текстом (например, через C-w) можно проделать то же, что и с текстом, вырезанным мышью. Поэтому вы можете вырезать несколько слов из вашего буфера Emacs и вставить их в xterm.

Правая кнопка мыши работает немного необычно. Если выделяется текст левой кнопкой мыши, то скопировать его можно, щелкнув один раз правой кнопкой. Второй щелчок правой кнопкой мыши удаляет текст. Чтобы вставить его обратно, нажмите среднюю кнопку мыши.[1] Текст будет вставлен прямо перед символом, на котором находится указатель мыши. Ошиблись? Ничего страшного, со всеми бывает; команда undo вернет все, как было, так же как и в случае всех остальных функций Emacs. (Выберите пункт Undo из меню Edit или просто нажмите C-_.)

Если вам по душе работа с мышью, можно привязать к кнопкам любые нужные вам функции точно так же, как и к клавиатуре. Попробуйте вставить в файл .emacs следующую команду:

```
(define-key global-map [S-mouse-1] 'mail)
```

Теперь, если нажать левую кнопку мыши, удерживая при этом клавишу Shift, должен появиться буфер создания сообщения электронной почты.

Мы не рекомендуем переопределять существующие функции мыши, но массу неиспользованных возможностей предлагают клавиши Shift, Ctrl и Alt. В привязках можно комбинировать S-, C- и M- любым образом:

```
(define-key global-map [S-C-mouse-1] 'mail)
```

Теперь давайте поиграем с окнами. Окна применялись в Emacs задолго до появления системы X Window. Поэтому окна Emacs – это не то же самое, что окна X Window. То, что в X считается окном, Emacs называет *фреймом* (*frame*).

Хотите редактировать один и тот же файл сразу в двух фреймах? Нажмите C-x 5 2, и появится другой фрейм. Новый фрейм – это просто еще одно представление редактируемого документа. Вы можете редактировать разные буферы в двух фреймах, но все, что вы сделаете в одном фрейме, будет повторено в соответствующем

[1] Если мышь без средней кнопки, возможно, была настроена эмуляция средней кнопки одновременным нажатием левой и правой кнопок. Кроме того, нажатие на колесико также может восприниматься как нажатие на среднюю кнопку.

буфере другого фрейма. Когда вы выйдете из Emacs, нажав C-x C-c, исчезнут оба фрейма; если вы хотите закрыть только один фрейм, нажмите C-x 5 0.

В завершение нашего исследования Emacs в системе X Window посмотрим на возможности цветового оформления. Изменять его можно прямо в сессии Emacs, что делает достаточно простыми эксперименты с различными возможностями. Нажмите M-x, затем наберите set-background-color и нажмите Enter. В ответ на запрос наберите ivory или любой другой цвет. (Помните, что Emacs использует соглашение M-x там, где в книге мы используем Alt+X.)

Постарайтесь выбрать цвета текста и фона так, чтобы они достаточно отличались и можно было бы видеть текст! Помимо set-background-color Emacs предлагает команды set-foreground-color, set-cursor-color и set-mouse-color.

Прежде чем завершить этот раздел, хочется отметить, что если Emacs или XEmacs кажутся вам слишком сложными, то будет приятно узнать, что в KDE входит целый ряд текстовых редакторов – от совсем простых до достаточно сложных. Ни один из них не обладает такими размерами или мощью, как (X)Emacs, но любого из них может оказаться достаточно.

Два основных текстовых редактора KDE называются KEdit и Kate. Последнее имя происходит от KDE Advanced Text Editor. Редактор Kate можно использовать как полнофункциональный редактор программиста, который поддерживает выделение синтаксиса цветом, может одновременно открывать несколько файлов и т. д. Редактор KEdit по богатству (или бедности) функциональных возможностей напоминает редактор Notepad в Windows. Существуют и другие редакторы, есть даже (вдохните глубже!) редактор *vi* с интерфейсом KDE. Все эти редакторы можно найти в K-меню, в подменю Editors (Редакторы). И наконец, если вы используете рабочий стол GNOME, определенно есть смысл попробовать поработать с редактором gedit.

20

Обработка текстовых документов

Зачем нужны анахронично выглядящие текстовые процессоры, описываемые в этой главе, когда весь мир пользуется текстовыми процессорами WYSIWYG, которые имеются даже в Linux? В действительности обработка текста (особенно в формате XML) – это то, что нас ждет в будущем. Пользователи будут стремиться к интерфейсам WYSIWYG, но на базе простого стандартного текстового формата, который сделает их документы переносимыми, предоставив при этом неограниченные возможности автоматизированной обработки документов.

Поскольку описываемые здесь средства являются программами с открытыми исходными текстами и широко доступны, можно пользоваться их форматами без зазрения совести и обоснованно предполагать, что у читателей ваших документов есть доступ к форматтерам. Для работы с этими форматами и выполнения сложной обработки, такой, например, как создание библиографии в TEX, вы также можете применять широкий спектр инструментальных средств, разрабатывавшихся годами. Наконец, разработаны фильтры, хотя и не всегда совершенные, для преобразования документов в этих форматах в другие популярные форматы, в том числе коммерческие, и обратно. Поэтому вы не оказываетесь полностью замкнутыми на эти форматы, хотя для точного преобразования может потребоваться некоторая ручная работа.

В первой главе мы кратко остановились на различных системах обработки текста, существующих в Linux, и их отличиях от текстовых процессоров, с которыми вы можете быть знакомы. В то время как большинство текстовых процессоров дают пользователю возможность вводить текст в среде WYSIWYG, системы обработки текста предлагают пользователю вводить исходный текст с использованием языка форматирования текста, который можно модифицировать в любом текстовом редакторе. (Фактически Emacs поддерживает специальные режимы редактирования для использования различных языков форматирования текста.) Затем исходный текст преобразуется в документ, пригодный для печати (или просмотра) с помощью собственно текстового процессора. Наконец, выходные данные обрабатываются и посылаются в файл или приложение, предназначенное для просмотра, либо передаются демону печати, который помещает их в очередь печати локального или удаленного устройства.

TeX и LaTeX

TEX – это профессиональная система обработки текста для всех типов документов, статей и книг, особенно тех, которые содержат много математических символов. Отчасти это язык «низкого уровня», поскольку он описывает, как система должна расположить текст на странице, какими должны быть промежутки и т. д. TEX напрямую не связан с такими элементами текста высокого уровня, как главы, разделы, сноски и т. д. (то есть тем, что нас, писателей, заботит больше всего). Поэтому TEX известен как функциональный язык форматирования текста (относящийся к реальному физическому расположению текста на странице), а не логический (относящийся к логическим элементам, таким как главы и разделы). TEX был разработан Дональдом Е. Кнутом, одним из крупнейших в мире специалистов по программированию. Одним из мотивов, побудивших его создать TEX, было стремление разработать систему набора текста, достаточно мощную для обслуживания потребностей форматирования математических выражений в написанной им серии книг по компьютерным наукам. Кнуту потребовалось восемь лет, чтобы закончить разработку TEX, и большинство согласно с тем, что результат стоит ожиданий.[1]

Конечно, TEX обладает большими возможностями расширения и допускает создание макросов, позволяющих авторам сосредоточиться преимущественно на логическом, а не физическом формате документа. И действительно, разработаны несколько таких макропакетов, наиболее популярным из которых является LATEX – набор расширений TEX, созданный Лесли Лэмпортом (Leslie Lamport). Команды LATEX касаются в основном логической структуры, но поскольку LATEX является просто набором макросов поверх TEX, он позволяет использовать и собственные команды TEX. LATEX значительно упрощает использование TEX, скрывая от пользователя большую часть функций низкого уровня.

Для создания хорошо структурированных документов с помощью TEX нужно либо использовать готовый пакет макросов, такой как LATEX, либо разработать свой собственный (или использовать комбинацию обоих). В книге «The TEX book»[2] Кнут представил собственный набор макросов, использовавшихся им при создании своей книги. Как и можно было ожидать, в их число входят команды для начала новых глав, разделов и т. п. Эти команды имеют сходство с соответствующими командами LATEX. В этом разделе мы сосредоточимся на использовании LATEX, который обеспечивает поддержку многих типов документов: технических статей, руководств, книг, писем и т. д. Как и TEX, LATEX допускает возможность расширения.

Как это работает

Если вы никогда раньше не работали с системой форматирования текста, тогда необходимо ознакомиться с рядом понятий. Как мы уже говорили, система обработки текста начинает с исходного текста, создаваемого с помощью обычного текстового редактора, такого как Emacs. Исходный текст создается на языке

---

[1] Дональд Кнут «Искусство программирования», т. 1–3, 3-е издание, «Все про METAFONT». – Пер. с англ. – М.: Вильямс, 2003; «Компьютерная типография». – Пер. с англ. – М.: Мир, 2003. – *Примеч. науч. ред.*

[2] Дональд Кнут «Все про TEX». – Пер. с англ. – М.: Вильямс, 2003.

форматирования текста, что означает ввод как текста, который должен появиться в документе, так и команд, сообщающих текстовому процессору, как он должен отформатировать этот текст.

Без дальнейших предисловий перейдем к практике. Напишем простой документ и отформатируем его от начала и до конца. Для иллюстрации мы покажем, как с помощью LaTeX написать короткое деловое письмо. Возьмите свой любимый редактор и введите следующий текст (конечно, без номеров строк). Сохраните его в файле *letter.tex*:

```
1  \documentclass{letter}
2  \address{755 Chmod Way \\ Apt 0x7F \\
3           Pipeline, N.M. 09915}
4  \signature{Boomer Petway}
5
6  \begin{document}
7  \begin{letter}{O'Reilly and Associates, Inc. \\
8                 1005 Gravenstein Highway North \\
9                 Sebastopol, C.A. 95472}
10
11 \opening{Dear Mr. O'Reilly,}
12
13 I would like to comment on the \LaTeX\ example as presented in
14 Chapter~20 of {\em Running Linux}. Although it was a valiant effort,
15 I find that the example falls somewhat short of what
16 one might expect in a discussion of text-formatting systems.
17 In a future edition of the book, I suggest that you replace
18 the example with one that is more instructive.
19
20 \closing{Thank you,}
21
22 \end{letter}
23 \end{document}
```

Это полный документ делового письма, которое мы хотим послать, в формате LaTeX. Как видите, в нем содержится фактический текст письма и ряд команд (с использованием символов обратного слэша и фигурных скобок). Разберем его последовательно.

В первой строке содержится команда `documentclass`, указывающая тип создаваемого документа (в данном случае `letter` – письмо). В LaTeX команды начинаются с символа обратного слэша, за которым следует действительное имя команды, в данном случае `documentclass`. За командой следуют аргументы, заключаемые в фигурные скобки. В LaTeX поддерживаются несколько типов документов, в том числе `article` (статья), `report` (отчет) и `book` (книга), а кроме того, вы можете определить собственный тип. Задание типа в документе TeX определяет глобальные макросы, которые будут использоваться в документе, такие как команды `address` и `signature`, используемые в строках 2–4. Как можно догадаться, команды `address` и `signature` указывают в письме ваш собственный адрес и имя. Два идущих подряд символа обратного слэша (`\\`) в адресе выполняют перевод строки в результирующем тексте адреса.

Несколько слов о том, как LaTeX обрабатывает входные данные. Как и в большинстве систем форматирования текста, пробелы, переводы строк и другие по-

добные элементы не передаются буквально в окончательный вывод, поэтому можно более или менее произвольно расставлять концы строк: при форматировании абзацев LaTeX снова соберет строки вместе. Есть, правда, исключения: пустые строки во входных данных начинают новые абзацы, и существуют команды, заставляющие LaTeX воспринимать исходный текст буквально.

В строке 6 команда *\begin{document}* используется для обозначения начала самого документа. Все, что заключено между *\begin{document}* и *\end{document}* в строке 22, считается частью форматируемого текста; все, что находится перед *\begin{document}*, называется преамбулой и задает параметры форматирования перед фактическим телом документа.

Само письмо начинается со строк 7–9 командой *\begin{letter}*. Это необходимо потому, что в одном исходном файле может быть много писем, и для каждого требуется *\begin{letter}*. Эта команда принимает в качестве аргумента адрес предполагаемого получателя. Как и в команде *address*, два идущих подряд символа обратного слэша обозначают переход на новую строку.

Строка 11 использует команду *opening* для открытия письма. За ней в строках 12–18 располагается фактическое тело письма. Как ни просто оно выглядит, но и в нем заключены некоторые особенности. В строке 13 команда `\LaTeX\` генерирует логотип LaTeX. Обратите внимание: символ обратного слэша ставится как перед командой `\LaTeX\`, так и после нее. Завершающий символ обратного слэша принуждает вставить пробел после слова «LaTeX». Это связано с тем, что TeX игнорирует пробелы после вызовов команд, поэтому за командой должны следовать символ обратного слэша и пробел. Так, например, команда `\LaTeX example` будет преобразована в текст `LaTeXexample`.

В строке 14 есть две детали, заслуживающие комментария. Прежде всего, это символ тильды (`~`), присутствующий между словами `Chapter` и `20`. Этот символ вызывает появление пробела между двумя словами, но предотвращает разрыв строки между ними в окончательном тексте (не допуская, таким образом, появления слова `Chapter` в конце строки и `20` — в начале следующей строки). Символ тильды нужно использовать только для обозначения пробела между двумя словами, которые должны располагаться на одной строке, как в `Chapter~20` и `Mr.~Jones`. (Оглядываясь назад, мы могли бы использовать тильду в командах *\begin{letter}* и *opening*, но маловероятно, что TeX разобьет строку внутри адреса или обращения.)

Второе, на что нужно обратить внимание в строке 14, это использование `\em` для создания выделенного текста в выводе. LaTeX поддерживает и другие шрифты, в том числе **boldface** (`\bf`) и `typewriter` (`\tt`).

В строке 20 письмо закрывается командой *closing*. Помимо этого она выполняет следующее действие: по окончании вывода приписывается подпись, определенная в строке 4. В строках 22–23 для завершения среды письма и документа, начатых в строках 6 и 7, используются команды *\end{letter}* и *\end{document}*.

Вы можете заметить, что команды в исходном тексте LaTeX совершенно не касаются таких функций, как установка полей, межстрочного интервала или других практических вопросов форматирования текста. Всем этим занимаются макросы LaTeX, работающие поверх ядра TeX. LaTeX устанавливает для этих параметров разумные значения по умолчанию. Если вам потребуется изменить какие-либо из этих параметров форматирования, можно использовать для этого другие команды LaTeX (или команды TeX низкого уровня).

Конечно, нельзя разобраться во всех сложностях использования LaTeX на таком ограниченном примере, но он должен дать представление о том, как выглядит живой документ LaTeX. Теперь давайте отформатируем документ, чтобы вывести его на печать.

Форматирование и печать

Можете не верить, но команда, используемая для превращения исходных текстов LaTeX в нечто печатаемое, называется *latex*. После редактирования и сохранения предыдущего примера в файле *letter.tex* вы можете выполнить команду

```
eggplant$ latex letter
This is pdfeTeX, Version 3.141592-1.21a-2.2 (Web2C 7.5.4)
(letter.tex
LaTeX2e <2003/12/01>
Babel <v3.8d> and hyphenation patterns for american, french, german, ngerman, bahasa,
basque, bulgarian, catalan, croatian, czech, danish, dutch, esperanto, estonian,
finnish, greek, icelandic, irish, italian, latin, magyar, norsk, polish, portuges,
romanian, russian, serbian, slovak, slovene, spanish, swedish, turkish, ukrainian,
nohyphenation, loaded.
(/usr/share/texmf/tex/latex/base/letter.cls
Document Class: article 2004/02/16 v1.4f Standard LaTeX document class
(/usr/share/texmf/tex/latex/base/size10.clo))
No file letter.aux.
[1] (letter.aux) )
Output written on letter.dvi (1 page, 1128 bytes).
Transcript written on letter.log.
eggplant$
```

latex предполагает, что исходные файлы имеют расширение *.tex*. В данном случае LaTeX обработал исходный текст *letter.tex* и сохранил результат в файле *letter.dvi*. Это независимый от устройства файл, который может быть выведен на ряд принтеров. Существуют различные утилиты преобразования файлов *.dvi* в PostScript, HP LaserJet и другие форматы, что мы вскоре покажем.

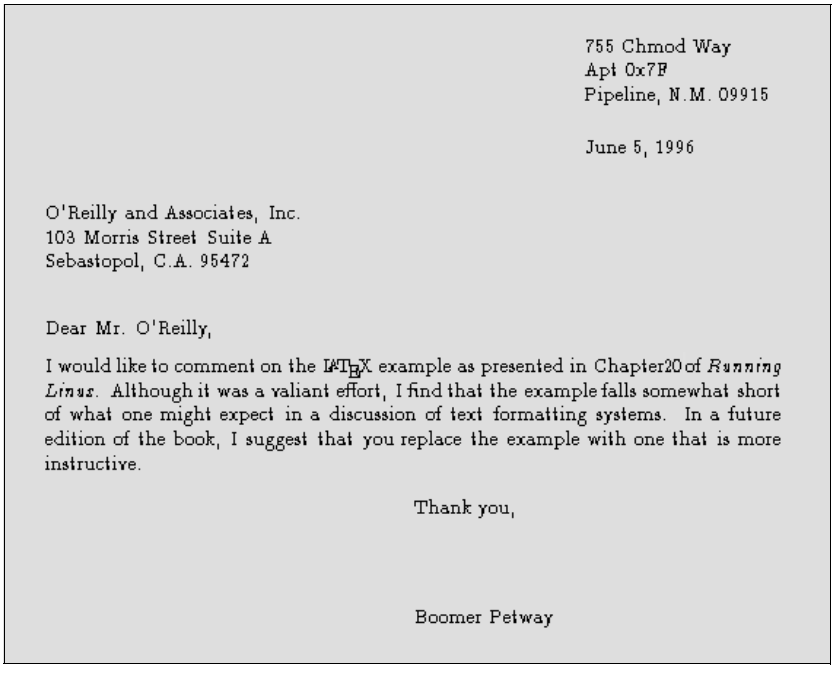
Прежде чем печатать письмо, вы, возможно, пожелаете его просмотреть, чтобы убедиться, что все выглядит нормально. Если вы работаете в системе X Window, то для предварительного просмотра файлов *.dvi* на экране можно использовать команду *xdvi*. Если у вас также стоит KDE, то там есть *kdvi*, которая является более дружественной версией *xdvi*. Так как же все-таки напечатать письмо? Прежде всего, необходимо конвертировать *.dvi* в формат, с которым может работать ваш принтер. Драйверы DVI существуют для многих типов принтеров. Почти всегда имена соответствующих программ начинаются с символов *dvi*, например *dvips*, *dvilj* и т. д. Если в вашей системе нет необходимого драйвера, нужно найти его в архивах в Интернете. Подробности вы найдете в FAQ для *comp.text.tex*.

Если вы счастливый обладатель принтера PostScript, то можете преобразовать *.dvi* в PostScript с помощью *dvips*:

```
eggplant$ dvips -o letter.ps letter.dvi
```

Затем можно вывести PostScript на печать с помощью команды *lpr*. Или сделать это за один шаг:

```
eggplant$ dvips letter.dvi | lpr
```



755 Chmod Way
Apt 0x7F
Pipeline, N.M. 09915

June 5, 1996

O'Reilly and Associates, Inc.
103 Morris Street Suite A
Sebastopol, C.A. 95472

Dear Mr. O'Reilly,

I would like to comment on the LaTeX example as presented in Chapter20 of *Running Linux*. Although it was a valiant effort, I find that the example falls somewhat short of what one might expect in a discussion of text formatting systems. In a future edition of the book, I suggest that you replace the example with one that is more instructive.

Thank you,

Boomer Petway

Рис. 20.1. *Пример вывода из файла*

Есть также драйверы для конкретных принтеров, например *dvilj* для HP LaserJet, но в большинстве своем они считаются устаревшими. Воспользуйтесь лучше *dvips* и при необходимости Ghostscript (об этом будет говориться ниже).

Кроме того, существует возможность заставить команду *dvips* отправить документ в формате PostScript непосредственно на принтер, например на принтер `lp`, как показано ниже:

```
eggplant$ dvips -Plp letter.dvi
```

Если вы не можете найти драйвер DVI для своего принтера, то, возможно, вам удастся с помощью Ghostscript преобразовать PostScript (полученный с помощью *dvips*) в какой-нибудь другой формат, который можно напечатать. Хотя некоторые шрифты Ghostscript далеко не оптимальны, он все же позволяет использовать шрифты Adobe (которые можно получить для Windows и использовать с Ghostscript под Linux). В Ghostscript есть также режим просмотра на SVGA, который можно использовать, если вы не работаете в X. Если вам удастся отформатировать и напечатать образец письма, оно должно выглядеть примерно так, как на рис. 20.1.

Наконец, нужно также отметить, что с помощью TeX можно создавать файлы PDF, используя для этого драйвер *dvipdf* или специальную программу *pdftex*.

Что еще прочесть

Если LaTeX подходит вам для обработки документов и вам удастся напечатать хотя бы этот начальный пример, мы советуем обратиться к книге Лесли Лэмпорта (Leslie Lamport) «LaTeX User's Guide and Reference Manual» (Addison Wesley), в которой описано все, что нужно знать о LaTeX для форматирования писем, статей, книг и прочего. Если вы интересуетесь программированием или хотите больше узнать о том, как работает TeX (что может оказаться очень ценным), то книга Дональда Кнута «The TeX book» (Addison Wesley) является исчерпывающим руководством по системе.

comp.text.tex является телеконференцией по этим системам, хотя данные, которые вы там обнаружите, предполагают, что у вас есть доступ к той или иной документации по TeX и LaTeX, например к отмеченным выше руководствам.

XML и DocBook

Язык разметки XML (и его предшественник SGML) делает еще один шаг вперед по сравнению с языками разметки, которые мы обсуждали до сих пор. Они накладывают на текст структуру, которая показывает отношение каждого элемента к содержащим его элементам. Это делает возможным преобразование текста в ряд выходных форматов, включая PostScript и PDF (Adobe Portable Document Format).

Язык XML сам по себе дает лишь базу для определения структуры документа. Так называемое описание типа документа (Document Type Definition, DTD) или схема устанавливают затем, какого рода разметку можно использовать в документе.

SGML – стандартный обобщенный язык разметки (Standard Generalized Markup Language) – был одним из первых стандартизированных языков, но в наши дни он практически вышел из употребления. Два его потомка – HTML и XML, вокруг которых возник даже излишний ажиотаж. По существу, HTML является реализацией SGML с фиксированным набором тегов, используемых для форматирования веб-страниц. XML является общим решением, как и SGML, но лишен некоторых сложных особенностей последнего. Как SGML, так и XML дают пользователям возможность определить любой набор тегов. Конкретно теги и их связи указываются в DTD или схеме (в XML это не обязательно).

Для каждого DTD или схемы, которые нужно использовать, необходимо иметь средства обработки, переводящие файл SGML или XML в нужный выходной формат. Исторически большинство бесплатных систем делали это с помощью системы под названием DSSSL (Document Style Semantics and Specification Language – язык описания семантики и стиля документа). Сейчас для преобразования XML в другие форматы гораздо популярнее XSLT (eXtensible Stylesheet Language Transformation – расширяемый язык таблиц стилей для преобразований). Как автора SGML- или XML-документа это совершенно не должно вас беспокоить, но если вы один из тех, кто занимается настройкой инструментальных средств, или вы собираетесь изменять внешний вид конечных документов, вам определенно потребуются знания о том, как все это происходит.

В области документации по компьютерам чаще всего используется DTD с названием DocBook. Помимо всего прочего, с помощью DocBook написана как большая часть бесплатной документации по Linux, так и эта книга. В число пользо-

вателей DocBook входит огромное число компаний и известных организаций, включая Sun Microsystems, Microsoft, IBM, Hewlett Packard, Boeing и Госдепартамент США.

Чтобы привести пример того, как может выглядеть текст DocBook, покажем фрагмент статьи для компьютерного журнала:

```
<!DOCTYPE Article PUBLIC "-//OASIS//DTD DocBook V4.1.2//EN">
<article>
  <artheader>
    <title>Looping the Froz with Foobar</title>
    <author>
      <firstname>Helmer B.</firstname>
      <surname>Technerd</surname>
      <affiliation>
        <orgname>Linux Hackers, Inc.</orgname>
      </affiliation>
    </author>
  </artheader>
  <abstract>
    <para>
      This article describes a technique that you can employ to
      loop the Froz with the Foobar software package.
    </para>
  </abstract>
  <sect1>
    <title>Motivation</title>
    <para>
      Blah, blah, blah, ...
    </para>
  </sect1>
</article>
```

В первой строке указаны используемое описание типа документа (DTD) и корневой элемент. В данном случае мы создаем статью с использованием DocBook DTD. Остальная часть исходного текста содержит саму статью. Если вы знакомы с HTML, языком разметки для WWW, текст должен показаться довольно знакомым. Теги используются для логической разметки текста. Все об HTML вы найдете в книге Чака Муссиано (Chuck Musciano) и Билла Кеннеди (Bill Kennedy) «HTML & XHTML: The Definitive Guide».[1]

Полное описание DocBook DTD выходит за рамки этой книги, но если вы заинтересовались, прочтите «DocBook: The Definitive Guide» Нормана Уолша (Norman Walsh) и Леонарда Мюльнера (Leonard Muellner), O'Reilly.

Когда вы напишете свою статью, документацию или книгу, то, конечно, захотите преобразовать их в формат, пригодный для печати или отображения на экране. Для того чтобы сделать это, вам потребуется полностью установить XML, что, к сожалению, сделать не так просто. В действительности это такая большая задача, что описать ее здесь невозможно. Но не все потеряно: ряд дистрибутивов,

[1] Чак Муссиано, Билл Кеннеди «HTML & XHTML. Подробное руководство», 6-е издание. – Пер. с англ. – СПб.: Символ-Плюс, 2008.

включая Red Hat, SUSE и Debian, поставляются с очень хорошими готовыми настройками XML; нужно просто установить соответствующие пакеты XML, которые в них входят. Если у вас есть работоспособная система SGML или XML, то можно преобразовать приведенный выше текст в HTML (как один из вариантов) с помощью такой команды:

```
tigger$ db2html myarticle.xml
input file was called -- output will be in myarticle
TMPDIR is db2html.C14157
working on /home/kalle/myarticle.xml
about to copy cascading stylesheet and admon graphics to temp dir
about to rename temporary directory to "myarticle"
```

В файле *myarticle/t1.html* будет находиться сгенерированный HTML-код. Если вместо этого требуется сгенерировать документ PDF, нужно выполнить такую команду:

```
tigger db2pdf myarticle.xml
tex output file name is /home/kalle/projects/rl5/test.tex
tex file name is /home/kalle/projects/rl5/test.tex
pdf file name is test.pdf
This is pdfeTeX, Version 3.141592-1.21a-2.2 (Web2C 7.5.4)
entering extended mode
(/home/kalle/projects/rl5/test.tex
JadeTeX 2003/04/27: 3.13
(/usr/share/texmf/tex/latex/psnfss/t1ptm.fd)
Elements will be labelled
Jade begin document sequence at 21
(./test.aux) (/usr/share/texmf/tex/latex/cyrillic/t2acmr.fd)
(/usr/share/texmf/tex/latex/base/ts1cmr.fd)
(/usr/share/texmf/tex/latex/hyperref/nameref.sty) (./test.out) (./test.out)
(/usr/share/texmf/tex/latex/psnfss/t1phv.fd) [1.0.49{/var/lib/texmf/fonts/map/p
dftex/updmap/pdftex.map}] [2.0.49] (./test.aux) ){/usr/share/texmf/fonts/enc/dvips/
psnfss/8r.enc}</usr/share/texmf/fonts/type1/urw/times/utmri8a.pfb></usr/share/texmf/
fonts/type1/urw/times/utmr8a.pfb></usr/share/texmf/fonts/type1/urw/helvetic/uhvb8a.pfb>
Output written on test.pdf (2 pages, 35689 bytes).
Transcript written on test.log.
```

Как видите, эта команда основывается на TEX, или, если говорить точнее, на его специальной версии под названием Jade, которая конкретно нацелена на документы, создаваемые DSSSL.

Это все хорошо, но если потребуется изменить внешний вид конечного документа, тогда использование DSSSL может оказаться весьма громоздким, не в последнюю очередь из-за нехватки доступной документации. Поэтому ниже будет дано краткое введение в современные механизмы обработки текстов на основе XSLT и FOP. Однако их использование почти всегда требует выполнения дополнительных действий по установке и настройке, включая чтение больших объемов документации.

Последовательность действий по обработке текста с применением XSLT выглядит следующим образом: сначала готовый документ XML плюс так называемая таблица стилей, написанная на языке XSL (eXtended Stylesheet Language – расширяемый язык таблиц стилей), обрабатываются процессором XSLT, например Saxon.

Язык XSL – еще один язык описания типа документа (DTD), или, другими словами, таблицы стилей документа XML. Она описывает, как должен обрабатываться каждый элемент в документе при преобразовании в другие элементы или в текст. Естественно, таблица стилей должна соответствовать описанию типа документа, в котором этот документ был создан. Кроме того, в зависимости от конечного формата представления документа могут потребоваться различные таблицы стилей.

Если конечным форматом является HTML, то на этом обработка документа заканчивается, потому что формат HTML сам по себе является подмножеством формата XML, и таблица стилей в состоянии преобразовать исходный документ XML в формат HTML, который может быть отображен в веб-броузере.

Если конечная цель далеко отстоит от формата XML (например, формат PDF), придется выполнить еще один шаг. Формат PDF не может быть получен непосредственно из документа XML, потому что формат PDF не является подмножеством формата XML, это скорее смешанный двоично-текстовый формат. Такое преобразование было бы очень сложно, если вообще возможно, выполнить только на основе XSLT. Поэтому здесь используется таблица стилей, которая генерирует XSL-FO – еще один акроним, начинающийся с X (eXtended Stylesheet Language Formatting Objects – расширяемый язык таблиц стилей объектов форматирования). Документ XSL-FO – это еще одна разновидность документов XML, в котором многие логические команды оригинального документа превращаются в физические команды.

Затем документ XSL-FO передается процессору FO, например Apache FOP, на выходе которого получается формат PDF (или любой другой формат, если он поддерживается процессором FO).

Теперь, когда в общих чертах обрисована картина происходящего, можно перейти к вопросу установки и настройки всего необходимого. Прежде всего, было бы неплохо проверить готовый документ XML с помощью программы проверки (валидатора). Программы такого рода не выполняют обработку документа, они лишь проверяют соответствие документа указанному DTD. Преимущество валидаторов состоит в том, что они выполняют проверку очень быстро. Поскольку фактическая обработка может занимать существенное время, будет совсем не лишним сначала узнать, насколько правильно оформлен документ, и в случае необходимости быстро исправить недостатки.

Примером такой программы проверки может служить *xmllint*. *xmllint* – это часть библиотеки *libxml2*, которая первоначально разрабатывалась для рабочего стола GNOME, но полностью независима от него (и фактически используется программным обеспечением рабочего стола KDE). Найти информацию о программе *xmllint* и загрузить ее можно на сайте *http://xmlsoft.org*.

xmlint используется следующим образом:

```
owl$ xmllint myarticle.xml > myarticle-included.xml
```

Причина, по которой *xmllint* выводит результаты своей работы на устройство стандартного вывода, состоит в том, что она может также использоваться для обработки X-Includes. Это методика, которая позволяет разделять документы XML на отдельные блоки удобным для авторов способом, а *xmllint* помогает снова объединить эти части. Дополнительную информацию о методике X-Includes можно найти на сайте *http://www.w3.org/TR/xinclude*.

Следующий шаг заключается в применении таблицы стилей. С этой задачей прекрасно справляется такой инструмент, как Saxon. Этот процессор распространяется в виде двух версий: написанной на языке Java или C++. Часто не имеет большого значения, какую версию устанавливать: версия, написанная на языке C++, работает быстрее, но версия, написанная на языке Java, обладает некоторыми дополнительными возможностями, такими как автоматическое экранирование служебных символов в подключаемых листингах программ. Получить дополнительную информацию и загрузить Saxon можно на сайте *http://saxon.sourceforge.net*.

Разумеется, для преобразования документа также потребуется таблица стилей (зачастую это набор большого числа файлов, ссылки на которые обычно находятся в единственном файле). Для работы с форматом DocBook достаточно будет установить пакет DocBook-XSL, который поддерживается светилами мира DocBook. Найти этот пакет можно на сайте *http://docbook.sourceforge.net/projects/xsl*.

Предположим, что используется версия процессора Saxon, написанная на языке Java, тогда команда создания документа XSL-FO могла бы выглядеть следующим образом:

```
java com.icl.saxon.StyleSheet myarticle-included.xml \
     docbook-xsl/fo/docbook.xsl > myarticle.fo
```

а для создания документа в формате HTML:

```
java com.icl.saxon.StyleSheet myarticle-included.xml \
     docbook-xsl/html/docbook.xsl > myarticle.html
```

Обратите внимание: выходной формат определяется только выбранной таблицей стилей.

Как уже говорилось ранее, если конечной целью является формат HTML, на этом работу можно считать выполненной. Для получения документа в формате PDF необходимо будет воспользоваться процессором FOP, таким как Apache FOP, который можно загрузить с сайта *http://xmlgraphics.apache.org/fop*. Для нормальной работы процессора FOP потребуется создать файл с настройками, как это сделать, описывается в документации. Часто в качестве файла с настройками можно использовать файл *userconfig.xml*, который поставляется вместе с FOP. Канонический вид команды вызова процессора для получения документа PDF приводится ниже:

```
java org.apache.fop.apps.Fop -c configfile myarticle.fo myarticle.pdf
```

Теперь вы имеете представление о том, как и какие инструментальные средства могут быть использованы. Помните, есть множество других аналогичных инструментов, которые могут удовлетворить ваши потребности более полно.

Вы можете спросить, в какой момент можно определить свои собственные требования к оформлению. До сих пор форматирование определялось таблицами стилей пакета DocBook-XSL. Это как раз и есть та точка, где вы можете включиться в процесс формирования конечного документа. Вместо того чтобы передавать процессору Saxon файл *docbook.xsl*, можно определить свой собственный файл. Разумеется, едва ли кто-нибудь захочет еще раз выполнить огромный объем работы, который был затрачен на создание DocBook-XSL, поэтому можно просто импортировать таблицу стилей DocBook-XSL в свою таблицу стилей и затем переопределить некоторые параметры. Ниже приводится пример такой таблицы стилей:

```
<?xml version='1.0'?>
<xsl:stylesheet xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
                xmlns:fo="http://www.w3.org/1999/XSL/Format"
                version='1.0'
                xmlns="http://www.w3.org/TR/xhtml1/transitional"
                exclude-result-prefixes="#default">

<xsl:import href="docbook-xsl/fo/docbook.xsl"/>
<xsl:param name="paper.type" select="'B5'"/>
<xsl:param name="shade.verbatim" select="1"/>
<xsl:param name="chapter.autolabel" select="1"/>

<xsl:attribute-set name="section.title.level1.properties">
  <xsl:attribute name="color">
    <xsl:value-of select="'#243689'"/>
  </xsl:attribute>
</xsl:attribute-set>

</xsl:stylesheet>
```

Что делает эта таблица? После типичного кода в самом начале элемент `<xsl:import>` загружает таблицу стилей генерации FOP по умолчанию (разумеется, здесь можно было бы использовать другую таблицу стилей для создания документа HTML). Затем выполняется переопределение некоторых параметров. Многие настройки в DocBook-XSL параметризованы, и потому элемент `<xsl:param>` – это все, что нам необходимо. В данном случае выбирается определенный формат бумаги, устанавливается затенение блоков цитат и определяется необходимость автоматической генерации меток оглавления для глав.

В заключение выполняется изменение цвета заголовков первого уровня, которое не может быть сделано простым изменением параметра. Здесь мы переопределяем атрибут цвета в наборе атрибутов. Более сложные изменения могут повлечь за собой необходимость полностью заменить определение элемента DocBook-XSL. Это далеко не простая задача, и потому мы советуем прочитать документацию DocBook-XSL от начала и до конца.

XML – это целый новый мир инструментов и технологий. Хорошей отправной точкой для вдохновения и его освоения может быть веб-сайт проекта Linux Documentation Project, который, как отмечалось ранее, использует формат XML/DocBook для всей своей документации. Вы обнаружите Linux Documentation Project на *http://www.tlpd.org*.[1]

groff

Параллельно TEX и независимо от него развивались *troff* и *nroff* – две другие значительные системы обработки текста, разработанные в Bell Labs для первоначальной реализации UNIX (фактически разработка UNIX была частично ускорена необходимостью поддержки такой системы обработки текста). Первая вер-

[1] Одно из обстоятельных и свежих русскоязычных руководств по достаточно объемной и путаной технике XML: *http://www.books.ru/shop/books/6228*. – *Примеч. науч. ред.*

сия этого текстового процессора называлась *roff* (от «runoff»). Затем появились *nroff* и *troff*, которые генерировали вывод для конкретных наборных машин, использовавшихся в то время. Система *nroff* была написана для принтеров с фиксированным шагом, например матричных принтеров, а *troff* – для устройств с пропорциональным шагом, первоначально для наборных машин. Более поздние версии *nroff* и *troff* повсеместно стали стандартными текстовыми процессорами на системах UNIX. *groff* является реализацией GNU *nroff* и *troff*, используемой в системах Linux. Она включает в себя несколько расширенных функций и драйверы для ряда печатающих устройств.

С помощью *groff* можно создавать документы, статьи и книги весьма схожим с TEX образом. Однако в *groff* (и первоначально в *nroff*) есть одна неотъемлемая функция, отсутствующая в TEX и его разновидностях: возможность выдавать результат в виде простого текста. В то время как TEX великолепен в случае создания документов для печати, *groff* может производить простой ASCII-текст, который можно просматривать на экране или печатать даже на простейших принтерах. Если вам нужно создавать документы, которые требуется и просматривать, и печатать, то, может быть, стоит обратить внимание на *groff* (хотя есть и другие альтернативы, в том числе программа Texinfo, о которой мы расскажем ниже).

Достоинство *groff* состоит и в значительно меньшем, по сравнению с TEX, размере: он требует меньше вспомогательных файлов и исполняемых модулей, чем минимальный дистрибутив TEX.

Одним из специальных применений *groff* является форматирование страниц справочного руководства UNIX. Если вы программируете в UNIX, то в конце концов вам понадобится писать и оформлять те или иные страницы справочного руководства. В этом разделе мы познакомим вас с *groff* на примере написания короткой страницы руководства.

Как и TEX, *groff* использует особый язык форматирования для описания того, как должен быть отформатирован текст. Этот язык несколько менее понятен, чем используемый в TEX, но и более лаконичен. Кроме того, есть несколько пакетов макросов для *groff*, выполняемых поверх базового форматтера *groff*. Эти пакеты макросов настроены на конкретный тип документа. Например, макросы `mgs` служат идеальным выбором при написании статей, а макросы `man` используются для создания страниц справочного руководства.

Создание страницы справочного руководства

С помощью *groff* очень просто создавать страницы справочного руководства. Для того чтобы ваша страница справочного руководства выглядела подобно другим, при написании исходного текста нужно соблюдать несколько правил, которые иллюстрируются следующим примером. Мы напишем страницу руководства для гипотетической команды *coffee*, которая может управлять сетевой кофеваркой.

Введите в текстовом редакторе следующий текст и сохраните его в файле *coffee.man*:

```
1  .TH COFFEE 1 "23 March 94"
2  .SH NAME
3  coffee \- Управление удаленной кофеваркой
4  .SH SYNOPSIS
```

```
5  \fBcoffee\fP [ -h | -b ] [ -t \fItype\fP ] \fIamount\fP
6  .SH DESCRIPTION
7  \fIcoffee\fP ставит в очередь запрос к удаленной кофеварке в устройстве
8  \fB/dev/cf0\fR. Обязательный аргумент \fIamount\fP задает
9  число чашек, обычно в диапазоне от 0 до 15, на кофеварках,
10 работающих в стандарте ISO.
11 .SS Options
12 .TP
13 \fB-h\fP
14 Варить горячий кофе. По умолчанию – холодный.
15 .TP
16 \fB-b\fP
17 Обжарить кофе. Особенно удобно при выполнении \fIcoffee\fP по поручению
18 вашего начальника.
19 .TP
20 \fB-t \fItype\fR
21 Указать сорт кофе для варки, \fItype\fP может иметь значение
22 \fBколумбийский\fP, \fBобычный\fP, \fBбез кофеина\fP.
23 .SH FILES
24 .TP
25 \fI/dev/cf0\fR
26 Устройство удаленной кофеварки
27 .SH "SEE ALSO"
28 молоко (5), сахар (5)
29 .SH BUGS
30 Может потребовать вмешательства оператора при израсходовании запаса кофе.
```

Пусть вас не пугает непонятность этого текста. Вам станет легче, когда вы узнаете, что последовательности символов `\fB`, `\fI` и `\fR` используются для изменения стиля шрифта на полужирный, курсив и прямой (roman), соответственно. `\fP` описывает возврат к шрифту, выбранному ранее.

Другие запросы *groff* появляются в строках, начинающихся с точки (`.`). В строке 1 запрос `.TH` устанавливает, что заголовком страницы руководства является `COFFEE`, а разделом руководства является 1. (Раздел 1 используется для команд пользователя, раздел 2 – для системных вызовов и т. д.) Запрос `.TH` устанавливает также дату последнего пересмотра страницы.

Запрос `.SH` в строке 2 начинает раздел, озаглавленный `NAME`. Обратите внимание: почти все страницы справочного руководства UNIX используют следующую последовательность разделов: `NAME`, `SYNOPSIS`, `DESCRIPTION`, `FILES`, `SEE ALSO`, `NOTES`, `AUTHOR` и `BUGS`. При необходимости включаются дополнительные разделы. Это всего лишь соглашение, используемое при составлении страницы, никак не требуемое программным обеспечением.

В строке 3 даются имя команды и, после дефиса (`\-`), ее краткое описание. Этот формат должен использоваться в разделе `NAME`, чтобы страницу руководства можно было ввести в базу данных *whatis*, используемую командами *man -k* и *apropos*.

В строках 4–5 дан краткий обзор синтаксиса команды *coffee*. Обратите внимание: для обозначения параметров командной строки используется курсивный шрифт (*\fI...\fP*), а необязательные параметры заключены в квадратные скобки.

В строках 6–10 дано краткое описание команды. Курсивный шрифт используется обычно для обозначения команд, имен файлов и параметров пользователя.

В строке 11 запрос .SS начинает подраздел с именем Options. Затем в строках 11–22 следует список параметров, представленный в виде маркированного списка. Каждый элемент списка помечен запросом .TP. Строка после .TP является тегом элемента, за которым следует непосредственно его текст. Например, исходный код строк 12–14:

```
.TP
\fB-h\fP
Варить горячий кофе. По умолчанию – холодный.
```

в итоге будет выглядеть как:

```
-h      Варить горячий кофе. По умолчанию – холодный.
```

Каждый параметр командной строки должен описываться аналогичным образом.

Строки 23–26 образуют раздел FILES страницы руководства, в котором описаны все файлы, которые могут быть использованы командой при работе. Для этого раздела также используется маркированный список с запросами .TP.

В строках 27–28 представлен раздел SEE ALSO, в котором даны перекрестные ссылки на другие страницы руководства, заслуживающие внимания. Обратите внимание, что строка "SEE ALSO" после запроса .SH в строке 27 заключена в кавычки. Это необходимо, поскольку .SH использует в качестве заголовка раздела первый аргумент, отделяемый пробелом. Поэтому все заглавия разделов, состоящие из нескольких слов, следует заключать в кавычки. Наконец, в строках 29–30 представлен раздел BUGS.

Форматирование и установка страницы руководства

Чтобы отформатировать эту страницу руководства и просмотреть ее на экране, выполните команду

```
eggplant$ groff  -Tascii -man coffee.man | more
```

Параметр *-Tascii* требует, чтобы *groff* создала вывод в виде простого текста; параметр *-man* требует, чтобы *groff* использовала набор макросов для страниц руководства. В случае успеха страница руководства будет выведена в следующем виде:

```
COFFEE(1)                                          COFFEE(1)

NAME
       coffee – Управление удаленной кофеваркой
SYNOPSIS
       coffee [ -h | -b ] [ -t type ] amount
DESCRIPTION
    coffee ставит в очередь запрос к удаленной кофеварке в устройстве
    /dev/cf0. Обязательный аргумент amount задает число чашек, обычно
    в диапазоне от 0 до 15 на кофеварках, работающих в стандарте ISO.

Options
    -h     Варить горячий кофе. По умолчанию – холодный.
    -b     Обжарить кофе. Особенно удобно при выполнении coffee
           по поручению вашего начальника.
    -t type
           Указать сорт кофе для варки, type может иметь
           значение «колумбийский», «обычный», «без кофеина».
```

```
FILES
    /dev/cf0
           Устройство удаленной кофеварки

SEE ALSO
    milk(5), sugar (5)

BUGS
    Может потребовать вмешательства оператора при израсходовании запаса кофе.
```

Как уже говорилось ранее, *groff* может производить другие типы документов. Использование параметра *-Tps* вместо *-Tascii* приводит к выводу результата в формате PostScript, который можно сохранить в файле, просмотреть с помощью Ghostview или вывести на принтер PostScript. Параметр *-Tdvi* порождает вывод в независимом от устройства формате *.dvi*, аналогичном производимому TEX.

Если вы хотите, чтобы страницу руководства могли видеть на вашей машине другие пользователи, нужно установить исходный текст для *groff* в каталоге, указанном в пользовательской переменной окружения `MANPATH`. Стандартные страницы руководства находятся в каталоге */usr/share/man*, хотя в некоторых системах используются также */usr/man* и */usr/local/man*. Поэтому текст страниц руководства раздела 1 должен находиться в */usr/man/man1*. Следующая команда:

```
eggplant$ cp coffee.man /usr/man/man1/coffee.1
```

установит эту страницу руководства для всеобщего использования в */usr/man* (обратите внимание на использование расширения *.1* вместо *.man*). Если после этого выполнить *man coffee*, то страница руководства будет автоматически переформатирована и текст для просмотра записан в */usr/man/cat1/coffee.1.gz*.

Если вам не разрешено копировать исходный текст страниц руководства прямо в */usr/man*, можно создать собственное дерево каталогов для страниц руководства и добавить его к своей переменной `MANPATH`. (Подробности вы найдете в разделе «Страницы справочного руководства» главы 4.)

Texinfo

Texinfo является системой форматирования текста, используемой в проекте GNU для создания электронной документации в виде гипертекстовых страниц Info и печатных материалов с помощью TEX из одного и того же файла исходного текста. Из исходного текста Texinfo пользователи могут преобразовывать документацию в любой из форматов: Info, HTML, DVI, PostScript, PDF или простой текст.

Документация по Texinfo составлена целиком через собственные страницы Info, которые можно читать в Emacs (используя команду `C-h i`) или с помощью отдельной программы чтения Info, такой как *info*. Если в вашей системе установлены страницы GNU Info, то в них содержится полная документация по Texinfo. Подобно тому как *groff* используется для составления страниц справочного руководства, Texinfo используется для составления документа Info.

Написание исходного кода Texinfo

В этом разделе мы покажем простой исходный файл Texinfo (по частям) и опишем, что делает каждый такой отрывок.

Наш исходный файл Texinfo будет называться *vacuum.texi*. Как обычно, исходный текст можно создать с помощью простого текстового редактора:

```
\input texinfo @c -*-texinfo-*-
@c %**start of header
@setfilename vacuum.info
@settitle The Empty Info File
@setchapternewpage odd
@c %**end of header
```

Это заголовок исходного текста Texinfo. Первая строка является командой TEX, используемой для ввода макросов Texinfo при создании печатной документации. Команды Texinfo начинаются с символа `@`. Команда `@c` служит началом комментария; комментарий `-*-texinfo-*-` является тегом, сообщающим Emacs, что это исходный текст Texinfo, благодаря чему Emacs может установить правильный основной режим. (Основные режимы обсуждались в разделе «Настройка Emacs» главы 19.)

Комментарии `@c %**start of header` и `@c %**end of header` служат для выделения заголовка Texinfo. Это требуется, если вы хотите отформатировать лишь часть файла Texinfo. Команда `@setfilename` указывает имя получаемого в результате файла Info, `@settitle` устанавливает заглавие документа, а `@setchapternewpage odd` сообщает Texinfo о необходимости начинать новые главы на нечетных страницах. Это стандартные процедуры, которые должны использоваться для всех файлов Texinfo.

Следующий раздел исходного файла задает титульный лист, используемый при форматировании документа с помощью TEX. Смысл этих команд должен быть очевиден:

```
@titlepage
@title Vacuum
@subtitle The Empty Info File
@author by Tab U. Larasa
@end titlepage
```

Теперь перейдем к телу исходного файла Texinfo. Файл Info делится на элементы, которые являются чем-то вроде страниц документа. Каждый элемент имеет ссылки на следующий, предыдущий и родительский элементы, а также может быть связан с другими элементами перекрестными ссылками. Каждый элемент можно представлять себе как главу или раздел документа с расположенным внизу меню для перехода к другим элементам. Например, элемент уровня главы имеет меню, перечисляющее разделы этой главы. Каждый элемент раздела указывает на элемент уровня главы, являющийся его родителем. Каждый раздел указывает также на предыдущий и последующий разделы, если они существуют. Это несколько запутанно, но станет ясным, когда вы увидите такую организацию в действии.

Каждому элементу дается короткое имя. Самый верхний элемент называется `Top`. Создание элемента начинается с команды `@node`, принимающей в качестве аргументов имя элемента, имя следующего элемента, имя предыдущего элемента и имя родительского элемента. Как было отмечено ранее, следующий и предыдущий элементы должны находиться на том же уровне иерархии. Родительский

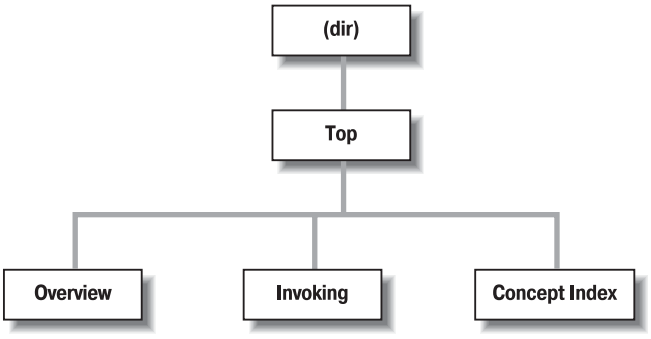


Рис. 20.2. *Иерархия элементов в Texinfo*

элемент находится над текущим в дереве элементов (например, родителем раздела 2.1 является глава 2). Пример иерархии элементов приведен на рис. 20.2.

Ниже приводится исходный текст элемента `Top`:

```
@c    Node, Next, Previous, Up
@node Top ,     ,          , (dir)

@ifinfo
Данный Info-файл является хорошим приближением к вакууму.
В нем не описывается абсолютно ничего.
@end ifinfo

@menu
* Overview::              Overview of Vacuum
* Invoking::              How to use Vacuum
* Concept Index::         Index of concepts
@end menu
```

Команде `@node` предшествует комментарий, напоминающий о порядке аргументов, передаваемых `@node`. В данном случае у элемента `Top` нет предыдущего и последующего элементов, поэтому эти аргументы оставлены пустыми. Родительским элементом для `Top` является `(dir)`, который указывает на общесистемный каталог страниц Info. Предполагается, что файл Info будет связан с системным деревом страниц Info, поэтому требуется, чтобы элемент `Top` имел обратную ссылку на общий каталог.

За командой `@node` следует обзор всего документа, заключенный в пару команд `@ifinfo...@end ifinfo`. Они используются для того, чтобы фактический текст элемента `Top` появлялся только в файле Info, но не в печатном документе, генерируемом $\TeX$.

Команды `@menu...@end menu` размечают меню элемента. Каждый пункт меню состоит из имени элемента, за которым следует его краткое описание. В данном случае меню указывает на узлы `Overview`, `Invoking` и `Concept Index`, исходный текст которых расположен дальше в файле. Эти три элемента являются тремя «главами» нашего документа.

Итак, продолжим. Рассмотрим элемент `Overview`, являющийся первой «главой»:

```
@c     Node,     Next,     Previous, Up
@node  Overview, Invoking,          , Top
@chapter Overview of @code{vacuum}

@cindex Nothingness
@cindex Overview
@cindex Vacuum cleaners

@code{vacuum} - это совершенно пустое пространство, лишенное любой материи. То есть
здесь нет ни воздуха, ни пива, ни пыли - вообще ничего нет. Обычно вакуум можно найти

в космосе. Пылесос - это устройство, создающее вакуум при чистке.
См. @xref{Invoking} о порядке использования @code{vacuum}.
```

Следующим элементом для `Overview` является `Invoking` — элемент второй «главы», который должен находиться в меню после `Overview`. В своих документах Texinfo вы можете использовать почти любые структуры, но часто бывает удобно организовать их так, чтобы узлы напоминали главы, разделы, подразделы и т. д. Решение остается за вами.

Команда *@chapter* начинает главу и имеет действие только при форматировании исходного текста в TEX. Аналогично команды *@section* и *@subsection* начинают разделы и подразделы в результирующем документе TEX. Имена глав, а также разделов и подразделов могут быть более описательными, чем короткие имена, используемые для самих элементов.

Обратите внимание на использование команды *@code* в названии главы. Это один из способов указать текст, который должен быть каким-то образом выделен. *@code* следует использовать для имен команд и исходного кода программ. Текст в команде *@code* выводится моноширинным шрифтом в TEX и заключается в одинарные кавычки (например, `'this'`) в файле Info.

Затем следуют три команды *@cindex*, создающие элементы указателя в конце документа. После этого появляется фактический текст элемента. И снова *@code* выделяет название «команды» *vacuum*.

Команда *@xref* порождает перекрестную ссылку на другой узел, к которому читатель может перейти командой *f* в программе чтения Info. *@xref* может производить перекрестные ссылки и на другие документы Texinfo. Полное изложение материала вы найдете в документации по Texinfo.

Следующим узлом является Invoking:

```
@node Invoking, Concept Index, Overview, Top
@chapter Порядок использования @code{vacuum}

@cindex Порядок использования @code{vacuum}
@code{vacuum} вызывается следующим образом:

@example
vacuum @var{options} @dots{ }
@end example
```

Здесь команды *@example*...*@end example* выделяют пример. Внутри примера *@var* обозначает метапеременную, замещающую строку, вводимую пользователем (в данном случае это параметры, передаваемые пользователем команде *vacuum*).

Команда `@dots{ }` порождает многоточие. В документе, отформатированном при помощи TEX, пример будет выглядеть так:

```
vacuum options …
```

а в файле Info так:

```
vacuum OPTIONS ...
```

Такие команды, как *@code* и *@var*, обеспечивают выделение, которое может осуществляться по-разному в TEX и Info.

Далее в элементе `Invoking` мы видим:

```
@cindex Options
@cindex Arguments
Поддерживаются следующие параметры:

@cindex Getting help
@table @samp
@item -help
Выводит краткое описание параметров.
@item -version
Выводит номер версии @code{vacuum}.

@cindex Empty vacuums
@item -empty
Производит особенно пустой вакуум. Используется по умолчанию.
@end table
```

Здесь описана таблица параметров, которые предположительно поддерживает команда *vacuum*. Команда *@table @samp* открывает таблицу из двух колонок (которая в итоге будет выглядеть, скорее, как маркированный список), каждый элемент которой будет выделен с использованием команды *@samp*. Команда *@samp* аналогична *@code* и *@var*, за исключением того, что предназначена для обозначения ввода литералов, таких как параметры командной строки.

Обычный документ Texinfo содержит элементы для примеров, сведений по предоставлению отчетов по ошибкам и многого другого, но для краткости мы закончим этот пример завершающим элементом — `Concept Index`. Это указатель понятий, встречающихся в документе, автоматически создающийся командой *@printindex*:

```
@node Concept Index, , Invoking, Top
@unnumbered Concept Index

@printindex cp
```

Команда *@printindex cp* сообщает форматтеру о необходимости вставить в это место указатель понятий. Есть и иные типы указателей, такие как указатель функций, указатель команд и прочие. Все они создаются с помощью разновидностей команд *@cindex* и *@printindex*.

Наконец, последние три строки нашего исходного текста Texinfo:

```
@shortcontents
@contents
@bye
```

Здесь форматтеру передаются команды для создания краткого оглавления (*@shortcontents*), полного содержания (*@contents*) и окончания форматирования (*@bye*). *@shortcontents* создает краткое оглавление, в котором перечислены только главы и приложения. Фактически только в случае длинных руководств в дополнение к *@contents* требуется *@shortcontents*.

Форматирование Texinfo

Чтобы создать Info-файл из исходного текста Texinfo, используется команда *makeinfo*. (Эта команда вместе с другими программами обработки Texinfo включается в дистрибутив программного обеспечения Texinfo, который иногда идет в связке с Emacs.) Команда

```
eggplant$ makeinfo vacuum.texi
```

создаст файл *vacuum.info* из *vacuum.texi*. *makeinfo* использует имя выходного файла, указанное в команде *@setfilename* в исходном коде. Его можно переопределить параметром *-o*.

Если результирующий файл оказывается большим, *makeinfo* разобьет его на несколько файлов с именами *vacuum.info-1*, *vacuum.info-2* и т. д., причем *vacuum.info* становится файлом верхнего уровня, указывающим на подчиненные файлы. Если все файлы *vacuum.info* находятся в одном каталоге, программа чтения Info сможет их найти.

Для создания Info из исходного текста Texinfo можно использовать также команды редактора Emacs `M-x makeinfo-region` и `M-x makeinfo-buffer`.

Теперь файл Info можно просматривать в Emacs, используя команду `C-h i`. В режиме Emacs Info необходимо будет использовать команду `g` и указать полный путь к файлу Info, например:

```
Goto node: (/home/loomer/mdw/info/vacuum.info)Top
```

Это связано с тем, что Emacs обычно ищет файлы Info только в собственном каталоге Info (которым может быть */usr/local/emacs/info*).

Альтернативой является использование *info* – независимой от Emacs программы чтения Info-файлов. Следующая команда:

```
eggplant$ info -f vacuum.info
```

запустит *info* на чтение вашего нового Info-файла.

Если вы хотите установить новую страницу Info для использования всеми пользователями системы, нужно добавить ссылку на нее в файл *dir* в каталоге Emacs info. В документации по Texinfo подробно описывается, как это сделать.

Чтобы создать из исходного текста печатный документ, нужно установить TEX в вашей системе. Программное обеспечение Texinfo поступает с файлом макросов для TEX – *texinfo.tex*, содержащим все макросы, необходимые для форматирования Texinfo с помощью TEX. При правильной установке *texinfo.tex* должен оказаться в каталоге *inputs*. Если это не так, скопируйте *texinfo.tex* в каталог, в котором находятся файлы Texinfo.

Сначала обработайте файл Texinfo с помощью TEX:

```
eggplant$ tex vacuum.texi
```

В результате в каталоге образуется множество файлов, из которых одни связаны с TEX, а другие используются при создании указателей. Команда *texindex* (входящая в состав пакета Texinfo) преобразует указатель в формат, который может использоваться TEX. Поэтому следующей должна быть выполнена команда

```
eggplant$ texindex vacuum.??
```

Использование `??` влечет обработку командой *texindex* всех файлов в каталоге, имеющих двухбуквенное расширение: это файлы, которые создаются Texinfo для генерации указателя.

Наконец, необходимо переформатировать файл Texinfo с помощью TEX, в результате чего удаляются перекрестные ссылки и вставляется указатель:

```
eggplant$ tex vacuum.texi
```

После выполнения этой команды вы должны получить файл *vacuum.dvi*, который можно просматривать с помощью *xdvi* или преобразовать в пригодный для печати вид. Ранее в этой главе в разделе «TEX и LATEX» обсуждалось, как печатать файлы *.dvi*.

Как обычно, осталось много нерассказанного об этой системе. Texinfo имеет полный набор собственных страниц Info, которые должны читаться вашей программой просмотра Info. Вы теперь также в состоянии отформатировать исходный текст документации по Texinfo с помощью TEX. Исходный текст документации по Texinfo – файлы *.texi* – должен быть в дистрибутиве исходного кода Texinfo.

21

Инструментальные средства программиста

Использование Linux предполагает значительно больше, чем простую работу в системе. Одно из преимуществ свободно распространяемого программного обеспечения состоит в том, что его можно модифицировать в соответствии со своими потребностями. Это в равной степени относится как к многочисленным свободно распространяемым программам для Linux, так и к самому ядру системы.

Linux поддерживает развитый интерфейс программирования, используя компиляторы и средства GNU, такие как компилятор *gcc*, отладчик *gdb* и другие. Поддерживается ряд языков программирования, начиная от классических FORTRAN и LISP и заканчивая современными языками сценариев, такими как Perl, Python и Ruby. Какой бы язык ни использовался, Linux является хорошей средой разработки приложений для UNIX. Поскольку доступны полные исходные тексты библиотек и ядра Linux, те программисты, которым необходимо покопаться во внутренностях системы, могут это сделать.[1]

Многие оценивают компьютерную систему по тому инструментарию, который она предоставляет программистам. По распространенному мнению, UNIX-системы выигрывают это соревнование, накопив с годами очень богатый набор таких средств. Возглавляет список отладчик GNU – *gdb*. В данной главе мы подробно познакомимся с этой ценной утилитой и рядом других вспомогательных средств, полезных для программистов, пишущих программы на языке C.

Даже если вы не программист, стоит познакомиться с системой контроля версий (Revision Control System, RCS). Она предоставляет одно из наиболее надежных средств защиты, о котором может мечтать пользователь компьютера, а именно создание файла с резервной копией всего, что вы делаете. Если вы случайно удалите файл или решите, что все ваши действия в течение последней недели были ошибкой и должны быть удалены, RCS поможет восстановить любую версию, которую вы пожелаете. Если вы работаете над большим проектом, в котором участвует большое число разработчиков или используется большое количество

---

[1] Во многих UNIX-системах авторы сталкивались с недостаточностью имеющейся документации. В Linux можно изучать сам исходный код ядра, библиотек и системных утилит. Возможность доступа к исходным текстам более важна, чем предполагает большинство программистов.

каталогов, то, возможно, больше подойдет система параллельных версий (Concurrent Versioning System, CVS). Изначально она была основана на RCS, но затем была переписана с самого начала и теперь содержит много дополнительных функций. В настоящее время начинает обретать популярность другой инструмент, с названием *Subversion*, который восполняет недостатки CVS при работе с крупными проектами.[1] Цель *Subversion* состоит в том, чтобы быть «как CVS, только лучше». Для разработки новых проектов все чаще выбирается *Subversion*, но огромное число существующих проектов все еще использует CVS. Наконец, для самого ядра Linux ранее использовалась еще одна система контроля версий под названием BitKeeper, но из-за возникших проблем с лицензированием Линус Торвальдс написал свою собственную систему управления версиями, получившую название *git*, которая была введена в строй совсем недавно.

Linux является идеальной платформой для разработки приложений, которые должны выполняться в среде X Window System. Дистрибутив Linux X, как отмечалось в главе 16, является полной реализацией со всем необходимым для разработки и поддержки приложений X. Само программирование для X позволяет получать переносимый программный код, поэтому ваши приложения в той части, которая относится к X, должны корректно компилироваться на других UNIX-системах.

В этой главе мы исследуем среду программирования Linux и дадим беглый обзор многочисленных средств, которые она предоставляет. Успех программирования в UNIX наполовину зависит от знания того, какие имеются инструменты и как их эффективно использовать. Часто самые полезные функции этих средств не очевидны новичкам.

Поскольку программирование на C лежит в основе большинства крупных программных проектов (хотя в настоящее время оно все более уступает программированию на языках C++ и Java) и C знаком большинству современных программистов не только в UNIX, но и в других операционных системах, мы начнем с рассказа о том, какие средства есть для этого языка. В первых разделах этой главы предполагается, что вы уже знакомы с языком программирования C.

Однако существуют и другие средства, которые приобретают все большее значение, особенно для системного администрирования. В этой главе мы рассмотрим одно из них – Perl. Perl – это язык сценариев, которые используются в оболочках UNIX. Он берет на себя тяжелую работу по управлению памятью, благодаря чему вы можете сосредоточиться непосредственно на своей задаче. Perl достаточно развит, что обеспечивает ему большую мощь, чем могут предоставить языки командной оболочки, и делает пригодным для решения различных задач программирования.

Некоторые проекты с открытыми исходными текстами реализованы на относительно легком и удобном языке программирования Java, а некоторые инструментальные средства и платформы, предназначенные для работы с этим языком, обрели в сообществе open source значительно более высокую популярность, чем те, что распространяются Sun Microsystems – компанией, которая разработала и ли-

[1] Название этого инструмента представляет собой весьма интересную игру слов, стоит только немного задуматься. («Subversion» можно перевести и как «диверсия, подрывная деятельность», и как «подверсия».)

цензировала язык Java. Java – это язык общего назначения с обширными возможностями для использования в Интернете. В одном из последующих разделов мы рассмотрим, что может предложить Java, и покажем, как начать с ним работать.

Программирование с использованием gcc

Язык программирования C используется значительно чаще других при разработке программ для UNIX. Возможно, это связано с тем, что сама система UNIX была первоначально разработана на C: это родной язык UNIX. Компиляторы C для UNIX традиционно определяют стандарты интерфейсов для других языков и средств, таких как компоновщики, отладчики и другие. Соглашения, предложенные в первоначальных компиляторах C, довольно последовательно соблюдаются во всех средствах программирования для UNIX.

Компилятор *gcc* является одним из наиболее гибких и развитых среди имеющихся компиляторов. В отличие от других компиляторов C (например, поставляемых в дистрибутивах AT&T или BSD либо сторонними разработчиками), *gcc* поддерживает все современные стандарты C, такие как стандарт ANSI C, а также собственные многочисленные расширения. К счастью, *gcc* совместим со старыми компиляторами C и старыми стилями программирования на C. Есть даже средство с названием *protoize*, позволяющее писать прототипы функций для программ C в старом стиле.

gcc является также компилятором C++. Для тех, кто предпочитает сложную объектно-ориентированную среду, поддерживается C++ со всеми его «художествами», включая возможности, появившиеся при выходе стандарта C++, такие как шаблоны методов. Имеются также полные библиотеки классов C++, такие как стандартная библиотека шаблонов (Standart Template Library, STL).

Для посвященных *gcc* предоставляет поддержку языка Objective-C – объектно-ориентированную разновидность C, которая не получила особого распространения, но обрела второе дыхание благодаря применению в Mac OS X. Кроме того, есть *gcj*, компилирующий код Java в машинный код. И этим, как мы увидим, дело не ограничивается.

В этом разделе мы рассмотрим использование *gcc* для компиляции и компоновки программ в Linux. Мы предполагаем, что вы знакомы с программированием на C/C++, но не предполагаем знакомства со средой программирования UNIX. Ее мы и собираемся здесь представить.

На момент написания книги последняя версия *gcc* имела номер 4.0. Однако она до сих пор считается достаточно новой и иногда проявляет свою нестабильность, а поскольку она более строго оценивает синтаксис исходных текстов, то может не компилировать более старый программный код. В связи с этим многие разработчики предпочитают пользоваться версией 3.3 (на момент написания этих строк текущей стабильной версией считалась версия 3.3.5) или 3.4. Мы предлагаем пользоваться одной из этих двух версий, если нет серьезных оснований для иного.

Забегая вперед, скажем несколько слов об используемой терминологии: современный компилятор *gcc* в состоянии компилировать исходные тексты, написанные

не только на C, но и на других языках программирования (например, C++, Java и некоторых других), таким образом, название *gcc* многими рассматривается как сокращение от «GNU Compiler Collection» (коллекция компиляторов GNU). Но если говорить исключительно о компиляторе языка C, тогда название *gcc* можно рассматривать как сокращение от «GNU C Compiler» (компилятор GNU языка C).

Краткий обзор

Прежде чем сообщить необходимые подробности собственно о *gcc*, мы приведем простой пример, показав на нем те шаги, через которые проходит компиляция программ на C в UNIX.

В качестве примера рассмотрим фрагмент кода избитой программы «Hello, World!»:

```
#include <stdio.h>
int main() {
    (void)printf("Hello, World!\n");
    return 0; /* Для приличия */
}
```

Чтобы скомпилировать эту программу в живой исполняемый программный код, нужно сделать несколько шагов. Большую часть их можно осуществить единственной командой *gcc*, но на подробностях мы остановимся в конце главы.

Прежде всего, компилятор *gcc* должен создать *объектный файл* из этого *исходного кода*. Объектный файл является, в сущности, машинным эквивалентом исходного текста на языке C. В нем содержится код для организации стека вызова `main()`, вызова функции `printf()` и код, возвращающий значение 0.

Следующим шагом является *компоновка* объектного файла с целью создания исполняемого модуля. Эту работу выполняет *компоновщик* (*linker*, или *редактор связей*). Его задача – взять объектные файлы, объединить их с программным кодом, содержащимся в библиотеках, и выдать исполняемый файл. Объектный код, полученный из исходного, не является законченной исполняемой программой. Прежде всего, нужно присоединить код функции `printf()`. Кроме того, к выполняемому модулю нужно присоединить некоторые подпрограммы инициализации, невидимые простому программисту.

Откуда берется код функции `printf()`? Из библиотек. Невозможно долго рассказывать о *gcc*, не упомянув библиотеки. Библиотека состоит, в сущности, из многочисленных объектных файлов и указателя на них. При поиске кода функции `printf()` компоновщик просматривает указатели всех библиотек, компоновку с которыми ему указано произвести. Он находит объектный файл, содержащий функцию `printf()`, извлекает этот объектный файл (весь объектный файл целиком, который может содержать значительно больше, чем одну функцию `printf()`) и присоединяет его к исполняемому модулю.

На самом деле все гораздо сложнее. Как упоминалось, в Linux есть два вида библиотек: *статические* (*static*) и *совместного доступа* (*shared*). В этом примере мы описывали статические библиотеки, то есть библиотеки, фактический код из которых для вызываемых подпрограмм присоединяется к исполняемому модулю. Однако код для таких подпрограмм, как `printf()`, может быть весьма объемным. Поскольку многие программы используют одни и те же библиотечные под-

программы, неразумно включать библиотечный код в каждый выполняемый модуль. Вот здесь на сцену и выходят библиотеки совместного доступа.[1]

При использовании *библиотек совместного доступа* код общих подпрограмм содержится в файле образа библиотеки на диске. При компоновке программы с библиотекой совместного доступа в исполняемый модуль вместо кода подпрограммы включается *заглушка* (*stub code*). Код, содержащийся в заглушке, сообщает загрузчику программы, где на диске в файле образа можно найти фактический код подпрограммы во время выполнения. Поэтому, когда запускается наша дружелюбная программа «Hello, World!», загрузчик обращает внимание, что программа скомпонована с библиотекой совместного доступа. Тогда он находит образ библиотеки совместного доступа и загружает код библиотечных функций, таких как `printf()`, вместе с кодом самой программы. Код заглушки сообщает загрузчику, где в файле образа находится код для `printf()`.

Даже такая картина является упрощением действительности. Библиотеки совместного доступа под Linux используют *таблицы переходов* (*jump tables*), позволяющие обновлять библиотеки и перемешивать их содержимое, при этом нет необходимости перекомпоновывать программы, использующие эти библиотеки. Код заглушки в исполняемом модуле фактически сам ищет другую ссылку в библиотеке, то есть в таблице переходов. Благодаря этому можно изменять содержимое библиотек и соответствующих таблиц переходов таким образом, что код заглушки может оставаться неизменным.

У библиотек совместного доступа есть еще одно преимущество – возможность обновления. Если кто-нибудь устранит программную ошибку в `printf()` (или брешь в системе безопасности, что важнее), вам нужно обновить лишь одну библиотеку, а не перекомпоновывать заново все программы в системе.

Но не позволяйте себе одурманиться этими абстрактными сведениями. В свое время мы возьмем реальный пример и покажем, как компилировать, компоновать и отлаживать программы. На самом деле это очень просто. О большинстве проблем позаботится вместо вас сам компилятор *gcc*. Однако неплохо понимать, что же происходит за кулисами.

Характеристики gcc

У *gcc* столько функций, что мы просто не сможем перечислить их в этой книге. Страница руководства по *gcc* и документ Info дадут вам массу интересных сведений об этом компиляторе. Далее в этом разделе мы дадим обстоятельный обзор наиболее полезных для начала функций *gcc*. С его помощью вы сами сможете определить, как заставить работать в ваших интересах и другие возможности.[2]

Для начинающих *gcc* поддерживает «стандартный» синтаксис языка C, используемый в настоящее время и определяемый в основном стандартом ANSI C. Са-

---

[1] Некоторые умные программисты считают, что библиотеки совместного доступа вредны. Основания для такого мнения слишком сложны, чтобы обсуждать их здесь. Они заявляют, что не стоит мучиться, когда на большинстве машин стоят диски по 80 Гбайт и не менее 256 Мбайт оперативной памяти.

[2] Есть русскоязычное издание полного руководства по GCC: Артур Гриффитс «GCC. Полное руководство. Platinum Edition». – Пер. с англ. – ДиаСофт, 2004. – *Примеч. науч. ред.*

мая важная особенность этого стандарта – прототипирование функций. Это означает, что функцию `foo()`, возвращающую значение типа `int` и принимающую два аргумента – `a` (типа `char *`) и `b` (типа `double`), можно определить следующим образом:

```
int foo(char *a, double b) {
    /* здесь находится ваш код... */
}
```

Это отличается от старого синтаксиса (без прототипов) определения функций, который выглядел так:

```
int foo(a, b)
char *a;
double b;
{
    /* здесь находится ваш код... */
}
```

Такой синтаксис тоже поддерживается *gcc*. Конечно, в ANSI C определены многие другие соглашения, но это наиболее заметное для начинающего программиста. Любой, кто знаком со стилем программирования на C по современным книгам, таким как второе издание «The C Programming Language»[1] Кернигана и Ритчи, без проблем сможет программировать с помощью *gcc*.

Компилятор *gcc* обладает развитыми механизмами оптимизации. В то время как большинство компиляторов C позволяют задавать для оптимизации только один ключ *-O*, *gcc* поддерживает различные уровни оптимизации. На высшем уровне оптимизации *gcc* совершает такие фокусы, как возможность совместного доступа к программному коду и статическим данным. Это означает, что если в вашей программе есть статическая строка, такая как `Hello, World!`, и ASCII-кодировка этой строки случайно совпала с последовательностью команд в вашей программе, то *gcc* позволит хранить данные строки и соответствующий программный код по одному адресу. Неплохо?

Конечно, *gcc* позволяет включать в объектные файлы отладочную информацию, что помогает отладчику (а значит, и программисту) осуществлять трассировку программы. Компилятор вставляет в объектный файл маркеры, дающие возможность отладчику находить в скомпилированной программе нужные строки, переменные и функции. Поэтому при использовании отладчика, например *gdb* (о котором мы поговорим ниже), можно одновременно пошагово выполнять программу и видеть исходный текст.

Среди других приемов, имеющихся в арсенале *gcc*, следует отметить возможность одним щелчком сгенерировать программный код на языке ассемблера. Можно потребовать от *gcc* не компилировать исходный текст в машинный код, а остановиться на уровне языка ассемблера, который значительно легче понять. Это прекрасный способ изучить сложности программирования под Linux на языке ассемблера в защищенном режиме процессора: напишите некоторый код на языке C, заставьте *gcc* перевести его на язык ассемблера и изучайте то, что получилось.

---

[1] Брайан Керниган, Деннис Ритчи «Язык программирования C», 2-е издание. – Пер. с англ. – Вильямс, 2006.

В *gcc* входит собственный ассемблер (который может использоваться независимо от *gcc* и называется *gas*, хотя при этом исполняемый файл в Linux называется *as*, благодаря чему исключается конфликт имен файлов других компиляторов с языка ассемблера, которые имеют место в других UNIX-системах, таких как Solaris) на случай, если вам понадобится ассемблировать этот код на языке ассемблера. Можно встраивать код на языке ассемблера в исходный текст C, если нужно осуществить какие-то особые чудеса, но вы не хотите писать на чистом ассемблере.

Основы работы с gcc

Вероятно, вам уже не терпится узнать, как пользоваться всеми этими замечательными функциями. Важно знать, как эффективно пользоваться *gcc*, особенно новичкам в UNIX и C. Использование компилятора командной строки типа *gcc* существенно отличается от работы в такой среде, как Visual Studio или C++ Builder в операционной системе Windows. Несмотря на сходство в синтаксисе языка, методы, используемые для компиляции и сборки программ, различны.

Ряд интегрированных сред (IDE) есть сейчас и для Linux. В их число входят популярная свободно распространяемая среда разработки KDevelop, о которой будет говориться далее в этой главе. Для разработки программ на языке Java по популярности среди тех, кто предпочитает пользоваться интегрированной средой разработки, лидирует Eclipse[1] (*http://www.eclipse.org*).

Вернемся к нашему невинному примеру «Hello, World!». Как бы вы скомпилировали и собрали эту программу?

Сначала, конечно, нужно ввести исходный текст. Это делается с помощью текстового редактора, такого как Emacs или *vi*. Программист должен ввести исходный программный код и сохранить его в файле с именем, скажем, *hello.c*. (Как и большинство компиляторов C, *gcc* очень разборчив в отношении расширений имен файлов; благодаря расширениям он может различать исходные тексты C, исходные тексты ассемблера, объектные файлы и т. д. Для файлов, содержащих стандартный исходный код на языке C, следует использовать расширение *.c*.)

Чтобы скомпилировать и собрать программу в исполняемый модуль *hello*, нужно выполнить команду:

```
papaya$ gcc -o hello hello.c
```

и (если нет ошибок) в один прием *gcc* скомпилирует исходный код в объектный файл, скомпонует его с соответствующими библиотеками и выдаст исполняемую программу *hello*, готовую к запуску. Подозрительный программист может попытаться протестировать полученный результат:

---

[1] Eclipse – платформонезависимая интегрированная среда, позволяющая строить проекты на различных языках программирования, которая на сегодня считается очень перспективной. По ней есть несколько русскоязычных изданий: Эд Барнет «Eclipse IDE. Карманный справочник». – Пер. с англ. – Кудиц-Пресс, 2006; Э. Гамма, К. Бек «Расширения Eclipse. Принципы, шаблоны и подключаемые модули». – Пер. с англ. – КУДИЦ-ОБРАЗ, 2005. – *Примеч. науч. ред.*

```
papaya$ ./hello
Hello, World!
papaya$
```

Настолько дружественно, насколько можно ожидать.

Очевидно, что при выполнении этой единственной команды *gcc* было произведено немало невидимых снаружи действий. Прежде всего, *gcc* нужно было скомпилировать ваш исходный файл *hello.c* в объектный файл *hello.o*. Затем скомпоновать *hello.o* со стандартными библиотеками и создать исполняемый модуль.

По умолчанию *gcc* предполагает, что вы хотите не только скомпилировать указанные исходные файлы, но и скомпоновать их вместе (друг с другом и со стандартными библиотеками) для создания исполняемого модуля. Сначала *gcc* компилирует исходные файлы в объектные. Затем он автоматически вызывает компоновщик, чтобы склеить все объектные файлы и библиотеки в исполняемый модуль. (В действительности компоновщик является отдельной программой с именем *ld*, а не частью самого *gcc*, хотя можно сказать, что *gcc* и *ld* являются близкими друзьями.) *gcc* также знает о «стандартных» библиотеках, используемых большинством программ, и он сообщает *ld* о необходимости компоновки с ними. Можно, конечно, рядом способов переопределить это поведение по умолчанию.

Можно указать несколько имен файлов в одной команде *gcc*, но в больших проектах оказывается более удобным скомпилировать сразу несколько файлов и хранить объектные файлы. Если нужно скомпилировать исходный код в объектный файл, но воздержаться от компоновки, используйте ключ *-c*:

```
papaya$ gcc -c hello.c
```

В результате будет создан лишь файл *hello.o* и ничего больше.

По умолчанию компоновщик создает исполняемый модуль с неожиданным именем *a.out*. Это пережиток ранних реализаций UNIX, на котором не стоит останавливаться. С помощью ключа *-o* в *gcc* можно присвоить выходному модулю другое имя, в данном случае *hello*.

Использование нескольких исходных файлов

Следующим шагом в вашем исследовании *gcc* будет выяснение того, как компилируются программы из нескольких исходных файлов. Предположим, ваша программа состоит из двух исходных файлов – *foo.c* и *bar.c*. Естественно, вы используете один или несколько заголовочных файлов (например, *foo.h*), содержащих объявления функций, используемых обеими программами. Благодаря этому код в *foo.c* знает о функциях в *bar.c*, и наоборот.

Чтобы скомпилировать эти два файла и скомпоновать их вместе (а также с библиотеками), создав исполняемый модуль *baz*, можно выполнить команду:

```
papaya$ gcc -o baz foo.c bar.c
```

Это примерно эквивалентно трем командам:

```
papaya$ gcc -c foo.c
papaya$ gcc -c bar.c
papaya$ gcc -o baz foo.o bar.o
```

gcc действует в качестве удобного интерфейса к компоновщику и другим «скрытым» утилитам, вызываемым во время компиляции.

Конечно, компиляция программы с использованием нескольких исходных файлов в одной команде может потребовать много времени. Если, например, у вас в программе пять или более файлов с исходным кодом, то *gcc* должен поочередно перекомпилировать каждый из них, прежде чем скомпоновать их в исполняемую программу. Это может вызвать большой и непроизводительный расход времени, особенно если вслед за последней компиляцией вы изменили только один файл. Нет смысла перекомпилировать остальные исходные файлы, поскольку их объектные файлы не изменялись и сохранили актуальность.

Эту проблему решает использование администратора проекта, такого как *make*. О *make* мы поговорим позже в разделе «Файлы проектов».

Оптимизация

Потребовать от *gcc* оптимизировать ваш код при компиляции просто. Для этого нужно лишь использовать ключ *-O* в командной строке *gcc*:

```
papaya$ gcc -O -o fishsticks fishsticks.c
```

Как мы недавно отмечали, *gcc* поддерживает несколько уровней оптимизации. Если использовать *-O2*, а не *-O*, то будут включены «дорогостоящие» виды оптимизации, которые могут замедлить компиляцию, но, надо надеяться, значительно увеличат производительность вашей программы.

При работе с Linux вы могли обратить внимание на то, что ряд программ компилируется с ключом *-O6* (примером служит ядро Linux). Текущая версия *gcc* не поддерживает оптимизацию до уровня *-O6*, поэтому (в настоящее время) этот ключ эквивалентен *-O2*. Однако *-O6* иногда используется для совместимости с будущими версиями *gcc* и обеспечения использования наивысшего уровня оптимизации.

Включение отладочной информации

Ключ *-g* заставляет *gcc* включить отладочную информацию в скомпилированные объектные файлы. Это означает, что в объектный файл и конечный исполняемый модуль добавляются данные, которые позволят трассировать программу с помощью отладчика, такого, например, как *gdb*. Недостатком использования отладочной информации является то, что она значительно увеличивает размер получаемых объектных файлов. Ключ *-g* обычно следует использовать во время разработки и отладки и не применять при «окончательной» компиляции.

К счастью, включение отладочной информации совместимо с оптимизацией. Это означает, что можно без опасений использовать команду:

```
papaya$ gcc -O -g -o mumble mumble.c
```

Однако некоторые виды оптимизации, осуществляемые по *-O* или *-O2*, могут вызвать хаотическое поведение программы при работе под отладчиком. Обычно лучше не использовать одновременно ключи *-O* и *-g*.

Еще о библиотеках

Прежде чем расстаться с *gcc*, нужно сказать несколько слов о компоновке и библиотеках. Начнем с того, что создать собственные библиотеки легко. Если у вас есть набор часто используемых функций, вы можете создать набор соответствующих исходных файлов, откомпилировать их все в объектные файлы и сделать из объектных файлов библиотеку. Тем самым вы избавите себя от необходимости отдельно компилировать эти функции для каждой программы, в которой они используются.

Предположим, что у вас есть набор часто используемых функций, например, такой:

```
float square(float x) {
    /* Реализция функции square()... */
}

int factorial(int x, int n) {
    /* Реализция функции factorial()... */
}
```

и так далее (конечно, в стандартных библиотеках *gcc* имеются аналоги этих распространенных функций, и пусть вас не смущает их выбор в нашем примере). Допустим, исходный текст `square()` находится в файле *square.c*, а исходный текст `factorial()` – в файле *factorial.c*. Все просто, не правда ли?

Чтобы создать библиотеку, содержащую эти функции, нужно лишь скомпилировать все файлы с исходными текстами следующим образом:

```
papaya$ gcc -c square.c factorial.c
```

В результате получатся объектные файлы *square.o* и *factorial.o*. Теперь создадим из них библиотеку. Оказывается, библиотека является просто архивным файлом, создаваемым утилитой *ar* (близким аналогом *tar*). Назовем нашу библиотеку *libstuff.a* и создадим ее следующим образом:

```
papaya$ ar r libstuff.a square.o factorial.o
```

При обновлении такой библиотеки может понадобиться предварительно удалить старую библиотеку *libstuff.a*. Последним шагом является создание указателя для библиотеки, который позволит компоновщику находить в ней функции. Для этого используется команда *ranlib*:

```
papaya$ ranlib libstuff.a
```

Эта команда добавляет данные к самой библиотеке и не создает отдельного файла указателя. Оба шага выполнения, *ar* и *ranlib*, можно объединить, используя команду *s* утилиты *ar*:

```
papaya$ ar rs libstuff.a square.o factorial.o
```

Теперь у вас есть статическая библиотека *libstuff.a*, содержащая ваши функции. Для того чтобы компоновать с ней программы, нужно создать файл заголовков, описывающий содержимое библиотеки. Например, можно создать *libstuff.h* со следующим содержимым:

```
/* libstuff.h: подпрограммы библиотеки libstuff.a */
```

```
extern float square(float);
extern int factorial(int, int);
```

Каждый исходный файл, использующий функции из *libstuff.a*, должен содержать строку `#include "libstuff.h"`, как для стандартных файлов заголовков.

Теперь, когда у нас есть библиотека и файл заголовка, как компилировать программы, чтобы их использовать? Прежде всего, нужно поместить библиотеку и файл заголовка в такое место, где компилятор сможет их найти. Многие пользователи помещают личные библиотеки в каталог *lib* своего домашнего каталога, а личные включаемые файлы – в каталог *include*. Предположив, что так мы и сделали, скомпилируем гипотетическую программу *wibble.c* с помощью команды:

```
papaya$ gcc -I../include -L../lib -o wibble wibble.c -lstuff
```

Параметр *-I* предлагает *gcc* добавить каталог *../include* к пути поиска заголовочных файлов. Аналогично параметр *-L* указывает *gcc* на необходимость добавить *../lib* к пути поиска библиотек.

Последний аргумент командной строки, *-lstuff*, сообщает компоновщику о необходимости присоединения библиотеки *libstuff.a* (где бы она ни находилась в списке каталогов поиска библиотек). Предполагается, что имена файлов библиотек начинаются с *«lib»*.

Когда вы хотите, чтобы компоновка производилась с библиотеками, отличными от стандартных, используйте ключ *-l* в командной строке *gcc*. Например, если вы хотите использовать математические функции (определенные в *math.h*), следует добавить в конец команды *gcc* ключ *-lm*, в результате чего компоновка будет осуществляться с библиотекой *libm*. Обратите внимание: порядок следования параметров *-l* имеет существенное значение. Например, если в нашей библиотеке *libstuff* есть такие же функции, как в *libm*, то *-lm* нужно включать в командную строку после *-lstuff*:

```
papaya$ gcc -Iinclude -Llib -o wibble wibble.c -lstuff -lm
```

Это вынуждает компоновщик присоединять *libm* после *libstuff* для нахождения ссылок, которые не были найдены в *libstuff*.

Где *gcc* ищет библиотеки? По умолчанию поиск происходит в нескольких местах, главным из которых является */usr/lib*. Если взглянуть на содержимое каталога */usr/lib*, можно заметить, что в этом каталоге содержится много библиотечных файлов, причем имена одних оканчиваются на *.a*, а других – на *.so.version*. Файлы *.a* являются статическими библиотеками, такими как наша *libstuff.a*. Файлы *.so* являются совместно используемыми библиотеками, содержащими код, присоединяемый во время исполнения, а также код заглушки, необходимый компоновщику времени исполнения (*ld.so*) для определения местоположения библиотеки.

Во время исполнения загрузчик программ ищет образы совместно используемых библиотек в разных местах, в том числе в каталоге */lib*. Если заглянуть в каталог */lib*, можно увидеть файлы типа *libc.so.6*. Это файл образа, содержащий код совместно используемой библиотеки *libc* (одной из стандартных библиотек, с которыми компонуется большинство программ).

По умолчанию компоновщик пытается осуществлять компоновку с совместно используемыми библиотеками, однако в некоторых случаях используются ста-

тические библиотеки, например, когда в пути поиска нет библиотеки совместного доступа с указанным именем. Можно также явно потребовать от *gcc* компоновки со статическими библиотеками с помощью ключа *-static*.

Создание совместно используемых библиотек

Теперь, когда вы знаете, как создавать и использовать статические библиотеки, очень легко перейти к созданию совместно используемых библиотек. У совместно используемых библиотек есть ряд преимуществ. Они сокращают расход памяти, если используются более чем одним процессом, и уменьшают размер исполняемого модуля. Более того, они облегчают разработку: при применении совместно используемых библиотек и внесении изменений в библиотеку не требуется всякий раз перекомпилировать приложение. Перекомпиляция требуется только при внесении несовместимых изменений, например при добавлении к вызову новых аргументов или изменении размеров структуры.

Пока вы не начали делать все свои разработки с помощью совместно используемых библиотек, мы должны вас предупредить, что отлаживаться с ними несколько сложнее, чем со статическими библиотеками, поскольку у отладчика *gdb*, обычно используемого в Linux, есть некоторые проблемы с совместно используемыми библиотеками.[1]

Программный код, включаемый в совместно используемую библиотеку, не должен зависеть от положения. Это соглашение для объектного кода, которое позволяет использовать код в совместно используемых библиотеках. *gcc* создает независимый от положения код, если передать ему в командной строке ключей *-fpic* или *-fPIC*. Предпочтительнее использовать первый ключ, если только модули не выросли настолько, что таблица перемещаемого кода оказалась слишком мала. В этом случае компилятор выдает сообщение об ошибке, и приходится использовать ключ *-fPIC*. Повторим наш пример из предыдущего раздела:

```
papaya$ gcc -c -fPIC square.c factorial.c
```

После этого создать совместно используемую библиотеку уже совсем просто:[2]

```
papaya$ gcc -shared -o libstuff.so square.o factorial.o
```

Обратите внимание на ключ компилятора *-shared*. Создавать указатель, как в случае статических библиотек, не требуется.

Использовать нашу вновь созданную совместно используемую библиотеку еще проще. В команду компилятора не нужно вносить никаких изменений:

---

[1] Кроме того, программа, скомпонованная с динамической бибилиотекой, будет несколько медленнее, чем ее эквивалент, скомпонованный статически (главным образом за счет необходимости компилировать код динамической бибилиотеки в позиционно независимый код с параметрами *gcc*: *-fpic* и *-fPIC*). Эта разница обычно малосущественна, но может стать значимой для задач интенсивных вычислений, например в цифровой обработке сигналов. – *Примеч. науч. ред.*

[2] На заре Linux создание разделяемых библиотек было устрашающей задачей, которой побаивались даже знатоки. С появлением несколько лет назад формата объектных файлов ELF задача свелась к выбору подходящего ключа при компиляции. Дело явно идет к лучшему!

```
papaya$ gcc -I../include -L../lib -o wibble wibble.c -lstuff -lm
```

Вы можете спросить, что делает компоновщик, если имеются как совместно используемая библиотека *libstuff.so*, так и статическая библиотека *libstuff.a*. В таком случае компоновщик всегда выбирает совместно используемую библиотеку. Для использования статической библиотеки ее нужно точно назвать в командной строке:

```
papaya$ gcc -I../include -L../lib -o wibble wibble.c libstuff.a -lm
```

Очень полезным инструментом при работе с библиотеками совместного доступа является *ldd*. С помощью этой утилиты можно узнать, какие библиотеки совместного доступа использует исполняемая программа, например:

```
papaya$ ldd wibble
        linux-gate.so.1 => (0xffffe000)
        libstuff.so => libstuff.so (0x400af000)
        libm.so.5 => /lib/libm.so.5 (0x400ba000)
        libc.so.5 => /lib/libc.so.5 (0x400c3000)
```

Три поля в каждой строке – это имя библиотеки, полный путь к используемому экземпляру библиотеки и адрес, в который отображается библиотека в виртуальном адресном пространстве. Первая строка – это нечто таинственное, имеющее отношение к реализации загрузчика в Linux, и ее можно пока не принимать во внимание.

Если для некоторой библиотеки *ldd* сообщает `not found`, то дело плохо и запустить эту программу вы не сможете. Придется поискать эту библиотеку. Возможно, она входит в состав вашего дистрибутива, но от ее установки вы отказались, либо она находится на жестком диске, но загрузчик (часть системы, загружающая все исполняемые программы) не может ее найти.

В последнем случае попытайтесь найти библиотеку самостоятельно и определить, не находится ли она в нестандартном каталоге. По умолчанию загрузчик осуществляет поиск только в */lib* и */usr/lib*. Если у вас есть библиотеки, находящиеся в другом каталоге, создайте переменную окружения `LD_LIBRARY_PATH` и укажите в ней каталоги, разделив их двоеточиями. Если вы уверены, что все настроено правильно, а требуемую библиотеку все равно не удается найти, выполните команду *ldconfig* в качестве *root*, чтобы обновить кэш компоновщика.

Использование C++

Если вы предпочитаете объектно-ориентированный стиль программирования, то *gcc* обеспечивает полную поддержку языка C++, а также Objective-C. При программировании на C++ с использованием *gcc* нужно учитывать несколько обстоятельств.

Во-первых, имена исходных файлов C++ должны иметь расширение *.cpp* (используется чаще всего), *.C* или *.cc*. Это отличает их от файлов с исходным текстом на обычном языке C, имеющих расширение *.c*. Фактически существует возможность вынудить *gcc* компилировать файлы с расширением *.c* как файлы с исходными текстами на языке C++, для этого следует использовать параметр командной строки *-x c++*, но такой способ не рекомендуется, поскольку он может вводить в заблуждение.

Во-вторых, при компиляции программного кода на языке C++ вместо *gcc* нужно использовать сценарий командной оболочки *g++*. Сценарий *g++* вызывает *gcc* с рядом дополнительных параметров, в частности, определяющих компоновку со стандартными библиотеками C++. *g++* использует те же ключи и параметры, что и *gcc*.

Если вы не хотите использовать *g++*, нужно обеспечить компоновку с библиотеками C++, чтобы использовать базовые классы C++, такие как объекты ввода-вывода `cout` и `cin`. Убедитесь также, что вы действительно установили библиотеки C++ и заголовочные файлы. В некоторых дистрибутивах содержатся только стандартные библиотеки C. *gcc* сможет откомпилировать ваши программы C++, но при отсутствии библиотек C++ дело закончится ошибками компоновщика, если вы пытаетесь использовать стандартные объекты.

Файлы проектов

Работая с Linux, вам, вероятно, придется столкнуться с утилитой *make*, даже если вы не собираетесь заниматься программированием. Вам, вероятно, понадобится устанавливать патчи или перекомпилировать ядро, а для этого придется запустить *make*. Если повезет, то вам не придется пачкаться с файлами проектов (makefiles), но мы постарались сделать эту книгу полезной не только для счастливчиков. Поэтому в данном разделе мы разъясним тонкий синтаксис *make*, чтобы вас не пугали файлы проектов.

В некоторых примерах мы будем использовать текущий файл проекта для ядра Linux. В нем используется множество расширений из мощной версии GNU *make*, поэтому мы опишем некоторые из них наряду со стандартными функциями *make*. Хорошее введение в *make* дано в книге «Managing Projects with make»[1] Эндрю Орэма (Andrew Oram) и Стива Талботта (Steve Talbott), O'Reilly. Расширения GNU хорошо документированы в руководстве по GNU *make*.

Большинство пользователей считают, что *make* – это средство создания объектных файлов и библиотек из исходных текстов, а также создания исполняемых модулей из объектных файлов. В более общем смысле *make* является универсальной программой создания *целевых объектов* (*targets*) на основе *зависимостей* (*dependencies*). Целевым объектом может являться выполняемая программа, документ в формате PostScript или что-то другое. Исходными данными могут быть текст на языке C, текстовый файл для TEX и т. д.

Можно, конечно, писать простые сценарии оболочки для выполнения команд *gcc*, которые будут создавать выполняемую программу, но особенность *make* состоит в том, что она знает, какие объекты нужно создавать заново, а какие – нет. Объектный файл нужно компилировать заново, только если изменился соответствующий файл с исходными текстами.

Допустим, у вас есть программа, состоящая из трех исходных файлов на языке C. Если бы вы создавали исполняемую программу с помощью команды:

```
papaya$ gcc -o foo foo.c bar.c baz.c
```

[1] Для русскоязычного читателя: см., например, В. Солдатов «Make, Build, Autotools. Управление программными проектами». – Бином, 2006. – *Примеч. науч. ред.*

то при каждом изменении одного из исходных файлов все три файла подвергались бы перекомпиляции и повторной сборке в исполняемый модуль. Если изменения касаются только одного файла, то это бесполезная трата времени (что особенно плохо, если программа состоит из большого числа исходных файлов). На самом деле вам нужно перекомпилировать только один изменившийся исходный файл в объектный файл и заново скомпоновать все объектные файлы. *make* может автоматизировать этот процесс.

Что делает make

Основная задача *make* – позволить создавать целевой файл мелкими шагами. Если конечный исполняемый модуль создается из многих исходных файлов, можно, изменив один исходный файл, собрать исполняемую программу, не перекомпилируя все подряд. Чтобы обеспечить подобную гибкость, *make* ведет учет файлов, необходимых для создания целевого объекта.

Приведем пример тривиального файла проекта. Назовите его *makefile* или *Makefile* и поместите в один каталог с файлами исходного кода:

```
edimh: main.o edit.o
        gcc -o edimh main.o edit.o

main.o: main.c
        gcc -c main.c

edit.o: edit.c
        gcc -c edit.c
```

Этот файл создаст программу с именем *edimh* из двух исходных файлов с именами *main.c* и *edit.c*. В файлах проектов вы не ограничены программированием на языке C, команды могут быть любыми.

В файле имеется три записи. В каждой из них есть строка *зависимостей*, показывающая, как строится файл. Таким образом, первая строка говорит, что *edimh* (имя перед двоеточием) строится из двух объектных файлов *main.o* и *edit.o* (имена после двоеточия). Эта строка сообщает *make*, что она должна выполнить следующую строку с командой *gcc*, если изменится один из этих объектных файлов. Строки, содержащие команды, должны обязательно[1] начинаться с символов табуляции (не с пробелов).

Команда

```
papaya$ make edimh
```

выполняет строку с командой *gcc*, если не существует файл с именем *edimh*. Однако эта строка выполняется и в случае, если *edimh* существует, но один из объектных файлов имеет более позднее время создания. *edimh* мы называем *целевым объектом* (*target*). Файлы после двоеточия называются *зависимостями* (*dependents*), или *пререквизитами* (*prerequisites*).

---

[1] Это давняя традиция *make*, и ее решили оставить как «историческое наследие». На эту ошибку попадался рано или поздно любой практикующий программист. – *Примеч. науч. ред.*

Следующие две записи делают то же самое в отношении объектных файлов. Файл *main.o* создается, если он не существует или ассоциированный с ним исходный файл имеет более позднее время создания. *edit.o* строится из *edit.c*.

Как *make* узнает о том, что файл обновился? Она смотрит на временную метку, которую файловая система связывает с каждым файлом. Эти временные метки можно увидеть, выполнив команду *ls -l*. Поскольку точность временной метки составляет одну секунду, она достаточно надежно сообщает *make* о том, подвергался ли файл изменениям со времени последней компиляции или компилировался ли объектный файл со времени последней сборки выполняемого файла.

Давайте испытаем файл проекта и посмотрим, что он делает:

```
papaya$ make edimh
gcc -c main.c
gcc -c edit.c
gcc -o edimh main.o edit.o
```

Если мы отредактируем *main.c* и повторно выполним команду, то перестроены будут только необходимые файлы, что сэкономит часть времени:

```
papaya$ make edimh
gcc -c main.c
gcc - o edimh main.o edit.o
```

Порядок расположения этих трех записей в файле проекта не имеет значения: *make* определяет зависимости файлов и выполняет команды в нужном порядке. Запись для *edimh* делается первой для удобства, так как в результате *edimh* становится файлом, создаваемым по умолчанию. Другими словами, ввод команды `make` становится равносильным вводу команды `make edimh`.

Ниже приводится более пространный файл проекта. Попробуйте разобраться в том, что он делает:

```
install: all
        mv edimh /usr/local
        mv readimh /usr/local

all: edimh readimh

readimh: read.o edit.o
        gcc -o readimh main.o read.o

edimh: main.o edit.o
        gcc -o edimh main.o edit.o

main.o: main.c
        gcc -c main.c

edit.o: edit.c
        gcc -c edit.c

read.o: read.c
        gcc -c read.c
```

Прежде всего, мы видим целевой объект `install`. Он не создает никакого файла и называется *фиктивным объектом*, поскольку существует только для того, чтобы могли быть выполнены команды, перечисленные под ним. Но прежде чем

выполнится `install`, должен выполниться `all`, поскольку `install` зависит от `all`. (Напомним, что порядок записей в файле не имеет значения.)

Итак, обратимся к объекту `all`. Под ним нет команд (это вполне допустимо), но он зависит от *edimh* и *readimh*. Это уже реальные файлы: каждый из них является исполняемой программой. Поэтому *make* прослеживает зависимости в обратном направлении, пока не придет к файлам *.c*, которые уже ни от чего не зависят. После этого она старательно перестраивает каждый из целевых объектов.

Ниже приводится пример прогона этого файла (вам могут понадобиться привилегии суперпользователя, чтобы установить файлы в каталоге */usr/local*):

```
papaya$ make install
gcc -c main.c
gcc -c edit.c
gcc -o edimh main.o edit.o
gcc -c read.c
gcc -o readimh main.o read.o
mv edimh /usr/local
mv readimh /usr/local
```

Таким образом, этот файл проекта вызывает полную компиляцию и установку. Сначала строятся файлы, необходимые для создания *edimh*. Затем строится дополнительный объектный файл, необходимый для создания *readmh*. После создания этих двух исполняемых файлов определяется целевой объект `all`. Теперь *make* может перейти к созданию целевого объекта `install`, что означает перемещение двух исполняемых файлов к месту их окончательного расположения.

Во многих файлах проектов, включая те, которые используются при компиляции Linux, содержатся фиктивные объекты для осуществления некоторых стандартных действий. Например, файл проекта для ядра Linux содержит команды для удаления временных файлов:

```
clean: archclean
        rm -f kernel/ksyms.lst
        rm -f core `find . -name '*.[oas]' -print`
        .
        .
        .
```

и создания списка объектных и заголовочных файлов, от которых они зависят (это сложная, но важная задача; если изменяется файл заголовка, необходимо перекомпилировать файлы, имеющие ссылки на него):

```
depend dep:
        touch tools/version.h
        for i in init/*.c;do echo -n "init/";$(CPP) -M $$i;done > .tmpdep
        .
        .
        .
```

Некоторые из этих команд оболочки становятся довольно сложными. Мы рассмотрим команды файла проекта далее в этой главе в разделе «Множественные команды».

Некоторые синтаксические правила

Самым сложным в работе с файлами проектов, по крайней мере на этапе освоения, является соблюдение правильного синтаксиса. Честно говоря, синтаксис *make* довольно бестолковый. Если использовать пробелы там, где должны быть символы табуляции, или наоборот, то файл проекта не обработается должным образом. А сообщения об ошибках просто ставят в тупик. Поэтому постарайтесь запомнить следующие правила:

- Всегда ставьте перед командой символ табуляции, а не пробелы. И не используйте табуляцию в начале других строк.
- В любом месте строки можно поставить символ #, чтобы начать комментарий. Символы после решетки игнорируются.
- Символ обратного слэша в конце строки означает продолжение команды на следующей строке. Это применяется для длинных командных строк и в некоторых других случаях.

Теперь рассмотрим некоторые мощные функции *make*, образующие своего рода язык программирования.

Макросы

Когда имя файла или другая строка используются в файле проекта неоднократно, их можно включить в макрос, то есть сделать строкой, которую *make* «разворачивает» в другую строку. Например, в начале нашего тривиального файла проекта можно было поместить такое макроопределение:

```
OBJECTS = main.o edit.o

edimh: $(OBJECTS)
        gcc -o edimh $(OBJECTS)
```

make просто вставит `main.o edit.o` во все места, где указано `$(OBJECTS)`. Если вам нужно добавить к проекту другой объектный файл, укажите его в первой строке. В результате строка зависимости и командные строки будут соответствующим образом изменены.

Не забывайте круглые скобки при ссылке на `$(OBJECTS)`. Макросы похожи на переменные окружения, такие как `$HOME` и `$PATH`, но это не одно и то же.

Один макрос можно определить через другой, поэтому допустимы, например, такие выражения:

```
ROOT = /usr/local
HEADERS = $(ROOT)/include
SOURCES = $(ROOT)/src
```

В этом случае `HEADERS` превращается в каталог */usr/local/include*, а `SOURCES` — в */usr/local/src*. Если вы устанавливаете пакет в системе и не хотите, чтобы он попал в */usr/local*, просто выберите другой путь и измените строку, определяющую `ROOT`.

Кстати, использовать верхний регистр в макроопределениях не обязательно, но это является общепринятым соглашением.

Расширение GNU *make* позволяет делать дополнения к макроопределению. При этом вместо знака равенства используется `:=`, например:

```
DRIVERS = drivers/block/block.a

ifdef CONFIG_SCSI
DRIVERS := $(DRIVERS) drivers/scsi/scsi.a
endif
```

Первая строка является обычным макроопределением, назначающим DRIVERS имя файла *drivers/block/block.a*. Следующее определение добавляет к нему имя файла *drivers/scsi/scsi.a*. Но это происходит только тогда, когда определен макрос CONFIG_SCSI. В этом случае полное определение становится следующим:

```
drivers/block/block.a drivers/scsi/scsi.a
```

А как определить CONFIG_SCSI? Можно вставить его в файл проекта, назначив ему произвольную строку:

```
CONFIG_SCSI = yes
```

Однако может оказаться более удобным определить его в командной строке *make*:

```
papaya$ make CONFIG_SCSI=yes target_name
```

Одна из тонкостей использования макросов состоит в возможности оставить их неопределенными. Если они не определены, то вместо них подставляется пустая строка. При этом у вас появляется возможность задать макрос в качестве переменной окружения. Например, можно не определять в файле проекта CONFIG_SCSI, а сделать это в файле с настройками *.bashrc*, используемом оболочкой bash:

```
export CONFIG_SCSI=yes
```

Если вы используете командную оболочку *csh* или *tcsh*, определение можно вставить в файл *.cshrc*:

```
setenv CONFIG_SCSI yes
```

В результате при любой сборке CONFIG_SCSI будет определен.

Правила использования суффиксов и шаблонов

Для таких рутинных задач, как сборка объектного файла из исходных текстов, не хотелось бы указывать в файле проекта все отдельные зависимости. И не нужно. Компиляторы UNIX используют простой стандарт (компилирование файла с суффиксом *.c* для создания файла с суффиксом *.o*), а *make* имеет функцию, называемую *правилом суффиксов*, работающую с такими файлами.

Ниже приводится пример использования простого правила суффиксов для компиляции исходных файлов на языке C, которое можно вставить в файл проекта:

```
.c.o:
        gcc -c $(CFLAGS) $<
```

Строка .c.o: означает: «использовать файл с исходными текстами, имеющий расширение *.c*, для создания файла *.o*». CFLAGS является макросом, в который можно вставить нужные параметры компилятора, например *-g* для отладки или *-O* для оптимизации. Строка $< является хитрым способом указать исходный файл. Таким образом, при выполнении этой команды в это место будет вставлено имя вашего файла *.c*.

Ниже приводится простой пример использования правила суффиксов. В командной строке передаются параметры *-g* и *-O*:

```
papaya$ make CFLAGS="-O -g" edit.o
gcc -c -O -g edit.c
```

Фактически нет необходимости указывать это правило суффиксов в файле проекта, поскольку нечто аналогичное уже встроено в *make*. Даже переменная `CFLAGS` в ней определена, поэтому параметры компиляции можно задать, просто установив значение этой переменной. Файл проекта для сборки ядра Linux содержит такое определение с множеством параметров *gcc*:

```
CFLAGS = -Wall -Wstrict-prototypes -O2 -fomit-frame-pointer -pipe
```

Говоря о флагах компилятора, надо особенно упомянуть одну группу. Это параметр *-D*, используемый для задания символов в выражениях исходного текста. Поскольку в условных выражениях типа `#ifdefs` встречается масса различных символов, может потребоваться передавать в файл проекта множество таких параметров, как `-DDEBUG` или `-DBSD`. Если вы делаете это через командную строку *make*, не забывайте весь набор заключать в двойные кавычки или апострофы, поскольку необходимо, чтобы оболочка передала весь набор как один аргумент:

```
papaya$ make CFLAGS="-DDEBUG -DBSD" ...
```

GNU *make* позволяет воспользоваться так называемыми *правилами шаблонов*, которые еще удобнее, чем правила суффиксов. Правило шаблонов использует знак процента, означающий «любую строку». Поэтому исходные тексты на языке C можно компилировать с помощью такого правила:

```
%.o: %.c
        gcc -c -o $@ $(CFLAGS) $<
```

Здесь сначала указан выходной файл *%.o*, а затем после двоеточия – исходный файл *%.c*. Правило шаблонов аналогично обычной строке зависимости, но вместо точных имен файлов содержит знаки процента.

Мы видим строку `$<`, относящуюся к исходному файлу, а также строку `$@`, указывающую на выходной файл, поэтому вместо нее вставляется имя объектного файла *.o*. Обе строки являются встроенными макроопределениями; *make* определяет их при обработке каждой записи.

Другим часто используемым встроенным макроопределением является `$*`, которое ссылается на имя исходного файла без расширения. Поэтому если исходным файлом является *edit.c*, строка `$*.s` означает *edit.s* (исходный файл на языке ассемблера).

Рассмотрим полезную вещь, которую можно сделать с помощью правила шаблонов, но нельзя сделать с помощью правила суффиксов, а именно: добавить строку `_dbg` к имени выходного файла для того, чтобы потом было видно, что он скомпилирован с отладочной информацией:

```
%_dbg.o: %.c
        gcc -c -g -o $@ $(CFLAGS) $<

DEBUG_OBJECTS = main_dbg.o edit_dbg.o

edimh_dbg: $(DEBUG_OBJECTS)
        gcc -o $@ $(DEBUG_OBJECTS)
```

Теперь все объекты можно создавать двумя различными способами: с отладочной информацией и без нее. У них будут разные имена, поэтому их можно хранить в одном каталоге:

```
papaya$ make edimh_dbg
gcc -c -g -o main_dbg.o main.c
gcc -c -g -o edit_dbg.o edit.c
gcc -o edimh_dbg main_dbg.o edit_dbg.o
```

Множественные команды

В проект *make* могут включаться любые команды оболочки. Однако дело осложняется тем, что *make* выполняет каждую команду в отдельной оболочке, поэтому, например, не будет работать такой проект:

```
target:
        cd obj
        HOST_DIR=/home/e
        mv *.o $HOST_DIR
```

Ни команда *cd*, ни определение переменной `HOST_DIR` не оказывают эффекта на последующие команды. Все команды должны быть объединены в одну строку. Оболочка использует в качестве разделителя между командами точку с запятой, поэтому строка может выглядеть так:

```
target:
        cd obj ; HOST_DIR=/home/e ; mv *.o $$HOST_DIR
```

Еще одно изменение связано с тем, что переменная оболочки в команде должна начинаться с удвоенного символа доллара, что позволяет *make* отличить ее от макроса.

Файл будет легче читаться, если разбить команды на несколько строк с помощью символа обратного слэша; при этом *make* продолжает считать такие строки одной командой:

```
target:
        cd obj ; \
        HOST_DIR=/home/e ; \
        mv *.o $$HOST_DIR
```

Иногда файлы проектов *make* содержат собственные команды *make*, то есть являются рекурсивными проектами, например:

```
linuxsubdirs: dummy
        set -e; for i in $(SUBDIRS); do $(MAKE) -C $$i; done
```

Макрос `$(MAKE)` вызывает *make*. Есть несколько причин использовать вложенные *make*. Одна из них – это выполнение сборки в нескольких каталогах, как в данном примере (каждый из них должен содержать собственный файл проекта *make*). В других случаях макросы определяются в командной строке, что позволяет выполнять проекты с различными макроопределениями.

GNU *make* предлагает в качестве расширения оболочки другой мощный интерфейс. Можно выполнить команду оболочки и присвоить ее вывод макросу. Ряд примеров такого рода есть в файле проекта для ядра Linux, но мы приведем простой пример:

```
HOST_NAME = $(shell uname -n)
```

Эта команда присваивает сетевое имя вашей машины, возвращаемое командой *uname -n*, макросу HOST_NAME.

make предлагает ряд других соглашений, которые иногда полезны. Например, знак @ перед командой подавляет отображение выполняемых команд:

```
@if [ -x /bin/dnsdomainname ]; then \
    echo #define LINUX_COMPILE_DOMAIN \"`dnsdomainname`\"; \
else \
    echo #define LINUX_COMPILE_DOMAIN \"`domainname`\"; \
fi >> tools/version.h
```

По другому соглашению, дефис, помещенный перед именем команды, сообщает *make*, что обработка проекта должна быть продолжена, даже если команда не могла быть выполнена. Это полезно, если вы хотите продолжить работу после неудачи с командой *mv* или *cp*:

```
- mv edimh /usr/local
- mv readimh /usr/local
```

Включение других файлов проекта

Большие проекты часто разбивают на несколько файлов. Это облегчает использование различными проектами в различных каталогах общих объектов, особенно макроопределений. Строка

```
include filename
```

читает содержимое файла *filename*. В файле проекта ядра Linux можно увидеть такую строку:

```
include .depend
```

Заглянув в файл *.depend*, вы обнаружите множество записей – строк, определяющих зависимость объектных файлов от заголовочных. (Кстати, *.depend* может изначально отсутствовать и создаваться другой записью в проекте.)

Иногда строки include ссылаются на макросы, а не на имена файлов, например:

```
include ${INC_FILE}
```

В данном случае имя *INC_FILE* должно быть определено как переменная окружения или как макрос, что дает больше возможностей управлять именем включаемого файла.

Интерпретация сообщений make

Сообщения об ошибках, выводимые *make*, могут быть весьма загадочными, поэтому мы поможем вам разобраться с ними. Следующие пояснения относятся к большинству стандартных сообщений.

**** No targets specified and no makefile found. Stop.*

Обычно это означает, что в компилируемом каталоге нет файла проекта. По умолчанию *make* сначала ищет файл *GNUmakefile*; в случае неудачи она ищет *Makefile* и, наконец, *makefile*. Если ни одного из этих файлов не суще-

ствует, выводится данное сообщение об ошибке. Если по каким-либо причинам вы хотите использовать файл проекта с другим именем (или в другом каталоге), можете задать его с помощью параметра командной строки *-f*.

*make: *** No rule to make target 'blah.c', needed by 'blah.o'. Stop.*

Это означает, что *make* не может найти требуемый исходный объект (в данном случае *blah.c*), чтобы построить целевой (в данном случае *blah.o*). Как уже упоминалось, *make* сначала ищет исходный объект среди целевых объектов в файле проекта, а если таких нет, то ищет файл с именем исходного объекта. Если и его не существует, выводится это сообщение об ошибке. Обычно это означает, что не хватает каких-то файлов с исходными текстами или в файле проекта допущена опечатка.

**** missing separator (did you mean TAB instead of 8 spaces?). Stop.*

Текущие версии *make* достаточно дружественны к пользователю, чтобы спросить его, не совершил ли он распространенной ошибки: пропустил символ табуляции перед командой. В более старых версиях *make* вы получите лишь сообщение *missing separator*. В этом случае проверьте, действительно ли у вас есть табуляция перед каждой командой и нигде более.

Autoconf, Automake и другие инструменты генерации Makefile

Создание больших файлов проектов обычно является утомительной и долгой задачей, особенно если предполагается компиляция на нескольких платформах. Проект GNU предоставляет два средства, называемые *Autoconf* и *Automake*, которые нелегко освоить, однако, овладев ими, можно сильно упростить задачу создания переносимых файлов проектов. Кроме того, создание переносимых библиотек совместного доступа весьма облегчается благодаря использованию *libtool*. Эти инструменты могут находиться на компакт-диске вашего дистрибутива, в противном случае их можно загрузить с сайта *ftp://ftp.gnu.org/gnu/*.

С точки зрения пользователя, использование *Autoconf* заключается в запуске программы *configure*, которая должна поставляться с пакетом исходного кода, который вы хотите скомпилировать. Эта программа анализирует вашу систему и настраивает файлы проектов пакета соответственно вашей системе и настройкам. Перед тем как реально запустить сценарий *configure*, неплохо выполнить такую команду:

```
owl$ ./configure --help
```

Она выведет все ключи командной строки, воспринимаемые программой *configure*. Многие пакеты допускают различную установку – например, могут включаться разные модули, и выбрать их можно с помощью параметров *configure*.

С точки зрения программиста, вы пишете не файлы проекта, а файлы с именем *makefile.in*. В них могут присутствовать символы-заместители, которые будут заменены фактическими значениями при выполнении пользователем программы *configure*, что создает файлы проектов, выполняемые затем *make*. Кроме того, вы должны написать файл с именем *configure.in*, который описывает ваш проект и то, что нужно проверить на целевой системе. После этого *Autoconf* генерирует программу *configure* из этого файла *configure.in*. К сожалению, написать файл

configure.in слишком сложно, чтобы эту процедуру можно было здесь описать, но в пакете *Autoconf* содержится необходимая документация по этому поводу.

Создание файлов *makefile.in* остается утомительной и долгой задачей, но даже ее можно автоматизировать с помощью пакета *Automake*. Используя этот пакет, вы пишете не файлы *makefile.in*, а файлы *makefile.am*, синтаксис которых значительно проще, а текст короче. С помощью *automake* файлы *makefile.am* преобразуются в *makefile.in*, включаемые в дистрибутивы исходных текстов и, в свою очередь, преобразуемые в *makefiles* при настройке пакета для системы пользователя. Описание правил составления файлов *makefile.am* также выходит за рамки этой книги. Здесь следует воспользоваться документацией, имеющейся в пакете.

В настоящее время большинство пакетов, распространяемых в виде исходных текстов, используют для генерации файлов проектов инструменты *libtool/automake/autoconf*, но это не значит, что этот довольно сложный и запутанный метод – единственный имеющийся. Есть и другие средства генерации файлов проектов, например утилита *imake*, используемая для настройки X Window System.[1] Существует еще один инструмент, не обладающий мощью пакета *Autoconf* (хотя и позволяющий делать большую часть из того, что нужно для генерации файлов проектов), но чрезвычайно простой в применении – он даже может генерировать собственные файлы описаний с самого начала. Имеется в виду *qmake* в составе библиотеки Qt C++ GUI (можно загрузить с сайта *http://www.trolltech.com*).

Отладка с помощью gdb

Не принадлежите ли вы к тем программистам, которые высмеивают саму мысль об использовании отладчика для трассировки программного кода? Вы считаете, что если код слишком сложен, чтобы даже программист мог его понять, то не стоит жалеть программиста, когда обнаруживаются ошибки? Проходите ли вы мысленно по своему коду, используя увеличительное стекло и зубочистку? Вызываются ли ошибки зачастую единственным пропущенным символом, например использованием оператора =, когда подразумевается +=?

Возможно, вам следует познакомиться с *gdb* – отладчиком GNU. Знаете вы об этом или нет, но *gdb* – ваш друг. Он может локализовать неясные и трудно обнаруживаемые ошибки, которые приводят к дампам памяти, утечке памяти и странному поведению (как программы, так и программиста). Иногда невинно выглядящие ошибки в вашей программе могут совершенно расстроить ее работу, и без помощи отладчика, такого как *gdb*, отыскать корень проблемы становится почти невозможно, особенно если в программе сотни и более строк. В этом разделе мы на нескольких примерах продемонстрируем использование наиболее полезных функций *gdb*. Кроме того, имеется выпущенная Фондом свободного программного обеспечения книга «Debugging with GDB», написанная Ричардом Столменом (Richard M. Stallman), Роландом Пешем (Roland Pesch), Стэном Шебсом (Stan Shebs) и др.

gdb может отлаживать программы во время выполнения или искать причину аварии программы по дампу памяти. Программы, отлаживаемые во время выполнения, могут запускаться из самого *gdb* или отдельно. Это означает, что *gdb* может

[1] Проект X.org планирует выполнить переход на *Automake/Autoconf* со следующей версии.

подключиться к уже исполняющемуся процессу, чтобы исследовать его. Сначала мы обсудим, как отлаживать программы, запущенные внутри *gdb*, а затем перейдем к подключению к работающим процессам и изучению дампов памяти.

Трассировка программы

Нашим первым примером будет программа *trymh*, определяющая границы изображения в оттенках серого цвета. *trymh* принимает в качестве исходных данных графический файл, производит некоторые вычисления, основываясь на полученных данных, и выдает результат в виде другого графического файла. К сожалению, программа вызывает аварийную ситуацию сразу при запуске:

```
papaya$ trymh < image00.pgm > image00.pbm
Segmentation fault (core dumped)
```

Теперь мы могли бы с помощью *gdb* проанализировать полученный файл с дампом памяти, но в этом примере мы покажем, как трассировать программу во время выполнения.[1]

Прежде чем использовать *gdb* для трассировки исполняемого модуля *trymh*, необходимо обеспечить, чтобы он был скомпилирован с отладочной информацией (подробности в разделе «Языки программирования и утилиты» главы 1). Для этого нужно компилировать *trymh* с помощью *gcc* с ключом *-g*.

Обратите внимание: одновременное использование ключей *-O* (для оптимизации) и *-g* (для включения отладочной информации) допустимо, но не рекомендуется. Например, если в двух разных местах функции есть одинаковые строки, то *gdb* может неожиданно перескочить ко второму вхождению строки вместо ожидавшегося первого. Это происходит из-за того, что *gcc* объединил две строки в одну строку машинного кода, используемую в обоих случаях.

Некоторые виды автоматической оптимизации, осуществляемой *gcc*, могут сбивать с толку при использовании отладчика. Для отключения всякой оптимизации (даже той, что производится без указания ключа *-O*) используйте при компиляции *gcc* ключ *-O0* («дефис-О-нуль»).

Теперь можно запустить *gdb* для определения проблемы:

```
papaya$ gdb trymh
GNU gdb 6.3
Copyright 2004 Free Software Foundation, Inc.
GDB is free software, covered by the GNU GeneralPublic License, and you are
welcome to change it and/or distribute copies of it under certain conditions.
Type "show copying" to see the conditions.
There is absolutely no warranty for GDB. Type "show warranty" for details.
This GDB was configured as "i386-suse-linux".
(gdb)
```

(GDB является свободно распространяемым программным обеспечением, и вы можете свободно модифицировать и/или передавать его копии при некоторых условиях, для ознакомления с которыми введите «show copying». На GDB нет абсолютно никаких гарантий; подробности можно увидеть, введя «show warranty».)

---

[1] Примеры программ, приведенные в этом разделе, вы вряд ли где-нибудь будете запускать; они были написаны авторами исключительно в демонстрационных целях.

Теперь *gdb* ждет команду. (Команда *help* выводит сведения об имеющихся командах.) Первое, что мы хотим сделать, – это запустить программу, чтобы наблюдать за ее поведением. Однако, если мы немедленно выполним команду *run*, программа будет просто выполняться, пока не закончит работу или не приведет к аварийной ситуации.

Поэтому сначала необходимо установить *точку останова* (*breakpoint*) где-либо в программе. Точка останова – это то место в программе, где *gdb* должен остановиться и позволить нам посмотреть, как выполняется программа. Для простоты установим точку останова на первой строке кода, чтобы программа остановилась сразу после начала выполнения. Команда *list* выводит сразу несколько строк программного кода (количество можно изменять):

```
(gdb) list
12        main() {
13
14          FloatImage inimage;
15          FloatImage outimage;
16          BinaryImage binimage;
17          int i,j;
18
19          inimage = (FloatImage)imLoadF(IMAGE_FLOAT,stdin);
20          outimage = laplacian_float(inimage);
21
(gdb) break 19
Breakpoint 1 at 0x289c: file trymh.c, line 19.
(gdb)
```

Теперь точка останова установлена в строке 19 текущего исходного файла. В программе можно установить много точек останова. Точки останова могут быть условными (то есть срабатывать только при истинности некоторого выражения), безусловными, отложенными, временно отключенными и т. д. Можно установить точку останова на конкретной строке, конкретной функции, группе функций и массой других способов. Можно также при помощи команды *watch* установить *контрольную точку* (*watchpoint*), которая аналогична точке останова, но срабатывает при возникновении некоторого события, причем не обязательно на указанной строке исходного кода программы. Подробнее о точках останова и контрольных точках будет рассказываться далее в этой главе.

Теперь воспользуемся командой *run* для запуска программы. *run* принимает те же аргументы, которые вы передали бы в командной строке *trymh*; они могут включать в свой состав маску имени файла и перенаправление потока ввода-вывода, когда команда передается */bin/sh* для выполнения:

```
(gdb) run < image00.pgm > image00.pfm
Starting program: /amd/dusk/d/mdw/vis/src/trymh < image00.pgm > image00.pfm

Breakpoint 1, main () at trymh.c:19
19          inimage = (FloatImage)imLoadF(IMAGE_FLOAT,stdin);
(gdb)
```

Как и следовало ожидать, мы немедленно оказываемся в точке останова в первой строке кода. Теперь мы можем перехватить управление.

Наиболее полезными командами пошагового выполнения программы являются *next* и *step*. Обе команды выполняют следующую строку кода программы, за исключением того, что *step* входит во все вызовы функций в программе, а *next* переходит только к следующей строке кода в той же функции. *next* спокойно выполняет все вызовы функций, которые ей встречаются, но не входит внутрь этих функций для их изучения.

`ImLoadF` – это функция, которая загружает образ из дискового файла. Мы знаем, что с этой функцией проблем не будет (вам придется поверить, что это так), поэтому хотим перескочить через нее с помощью команды *next*:

```
(gdb) next
20          outimage = laplacian_float(inimage);
(gdb)
```

Здесь мы хотим отследить подозрительно выглядящую функцию `laplacian_float`, поэтому выполняем команду *step*:

```
(gdb) step
laplacian_float (fim=0x0) at laplacian.c:21
21          i = 20.0;
(gdb)
```

Выполним команду *list*, чтобы получить представление о том, где мы находимся:

```
(gdb) list
16        FloatImage laplacian_float(FloatImage fim) {
17
18          FloatImage mask;
19          float i;
20
21          i = 20.0;
22          mask=(FloatImage)imNew(IMAGE_FLOAT,3,3);
23          imRef(mask,0,0) = imRef(mask,2,0) = imRef(mask,0,2) = 1.0;
24          imRef(mask,2,2) = 1.0;  imRef(mask,1,0) = imRef(mask,0,1) = i/5;
25          imRef(mask,2,1) = imRef(mask,1,2) = i/5;  imRef(mask,1,1) = `i;
(gdb) list
26
27          return convolveFloatWithFloat(fim,mask);
28        }
(gdb)
```

Как видите, повторное выполнение *list* просто выводит следующий дальше программный код. Поскольку мы не хотим проходить этот код пошагово вручную и нас не интересует функция `imNew` в строке 22, продолжим выполнение до строки 27. Для этого выполним команду *until*:

```
(gdb) until 27
laplacian_float (fim=0x0) at laplacian.c:27
27          return convolveFloatWithFloat(fim,mask);
(gdb)
```

Прежде чем войти в функцию `convolveFloatWithFloat`, убедимся, что два параметра, `fim` и `mask`, имеют допустимые значения. Команда *print* показывает значение переменной:

```
(gdb) print mask
$1 = (struct {...} *) 0xe838
(gdb) print fim

$2 = (struct {...} *) 0x0
(gdb)
```

Параметр *mask* выглядит нормально, но *fim* (образ ввода) оказался пустым (null) указателем. Очевидно, что функции laplacian_float был передан пустой указатель вместо действительного образа. Если вы внимательно смотрели, то заметили это при входе в laplacian_ float.

Вместо того чтобы глубже залезать в программу (поскольку уже ясно, что возникли проблемы), продолжим выполнение до возврата из текущей функции. Это осуществляет команда *finish*:

```
(gdb) finish
Run till exit from #0 laplacian_float (fim=0x0) at laplacian.c:27
0x28c0 in main () at trymh.c:20
20          outimage = laplacian_float(inimage);
Value returned is $3 = (struct {...} *) 0x0
(gdb)
```

Теперь мы снова находимся в функции main. Чтобы определить источник проблемы, посмотрим значение некоторых переменных:

```
(gdb) list
15          FloatImage outimage;
16          BinaryImage binimage;
17          int i,j;
18
19          inimage = (FloatImage)imLoadF(IMAGE_FLOAT,stdin);
20          outimage = laplacian_float(inimage);
21
22          binimage = marr_hildreth(outimage);
23          if (binimage == NULL) {
24            fprintf(stderr,"trymh: binimage returned NULL\en");
(gdb) print inimage
$6 = (struct {...} *) 0x0
(gdb)
```

Переменная inimage, содержащая входной образ, возвращаемый функцией imLoadF, имеет нулевое значение. В таком случае передача нулевого указателя в процедуры обработки изображения несомненно вызовет аварийный останов с выводом дампа памяти. Однако нам известно, что imLoadF проверена и правильна, поскольку находится в хорошо протестированной библиотеке. Так в чем же дело?

Оказывается, наша библиотечная функция imLoadF возвращает значение NULL в случае неудачи, например при неверном формате входного файла. Поскольку мы не проверяли значение, возвращаемое imLoadF, прежде чем передать его функции laplacian_float, программа приходит в расстройство, когда переменной inimage присваивается значение NULL. Для решения проблемы мы просто вставим код, вызывающий выход с сообщением об ошибке, когда imLoadF возвращает пустой указатель.

Для выхода из *gdb* выполните команду *quit*. Если программа не закончила выполнение, *gdb* сообщит, что программа все еще выполняется:

```
(gdb) quit
The program is running. Quit anyway (and kill it)? (y or n) y
papaya$
```

В следующих разделах мы рассмотрим некоторые специальные функции, предоставляемые отладчиком в ситуации, с которой мы столкнулись.

Исследование файла дампа памяти

Нравится ли вам, когда программа не только аварийно завершается, но еще оставляет в рабочем каталоге 20-мегабайтный файл дампа памяти, при том что рабочее пространство так необходимо? Не спешите удалять этот файл; он может быть очень полезен. Файл дампа памяти является копией образа памяти процесса в тот момент, когда произошла авария. Вы можете использовать этот файл с *gdb* для изучения состояния вашей программы (например, значений переменных и данных) и определения причины отказа.

Файл дампа памяти записывается на диск операционной системой, когда происходят некоторые виды отказов. Наиболее частой причиной отказа и последующего вывода дампа памяти является нарушение границ памяти, то есть попытка чтения или записи по адресам, к которым у вашей программы нет права доступа. Например, попытка записи по нулевому адресу может вызвать *ошибку сегментации* (в этом случае сразу ясно, что вы «напортачили»). Ошибки сегментации являются частыми ошибками и возникают при попытке доступа (для записи или чтения) по адресу, не принадлежащему адресному пространству вашего процесса. В число этих адресов входит нулевой адрес, который часто возникает при пропуске инициализации указателя. Ошибка сегментации часто вызывается попыткой обращения к элементу массива, находящемуся за границами объявленного размера массива, обычно при превышении границы на единицу. Эти ошибки могут также быть обусловлены тем, что не была выделена память для некой структуры данных.

Другие ошибки, приводящие к созданию дампа памяти, называются «ошибками шины» и «ошибками операций с плавающей запятой». Ошибки шины происходят при неправильном выравнивании данных и поэтому редко встречаются в архитектуре Intel, не предъявляющей таких строгих условий выравнивания, как другие архитектуры. Исключения, возникающие при выполнении операций с плавающей запятой, свидетельствуют о серьезных проблемах вычислений с плавающей запятой, таких как переполнение, но чаще всего ошибка состоит в делении на нуль.

Однако не все такие ошибки приводят к немедленному краху программы. Например, вы можете каким-то образом залезть в неверную область памяти, но программа продолжит выполнение, не видя разницы между фактическими данными или командами и мусором. Тонкие нарушения границ памяти могут вызвать неустойчивое поведение программ. Один из авторов был свидетелем случайных переходов в программе, но до трассировки ее с помощью *gdb* она казалась нормально работающей. Единственным признаком наличия ошибки в программе был ее вывод, показывающий, грубо говоря, что два плюс два не равно

четырем. Оказалось, что ошибка состояла в попытке записать лишний символ в отведенный блок памяти. Ошибка в один байт вызвала долгие часы страданий.

Избежать проблем такого рода (даже хорошие программисты совершают такие ошибки!) можно с помощью пакета Valgrind – набора программ управления памятью, заменяющих часто используемые функции `malloc()` и `free()`, а также их аналоги в C++ – операторы `new` и `delete`. Мы обсудим Valgrind в разделе «Использование библиотеки Valgrind», далее в этой главе.

Однако, если ваша программа производит нарушение границ памяти, она вызовет аварийный останов и дамп памяти. В Linux файлы памяти так и называются – *core*. Файл *core* появляется в текущем рабочем каталоге исполняемого процесса, которым обычно является текущий каталог оболочки, запустившей программу, но иногда программы могут изменить свой рабочий каталог.

Некоторые оболочки предоставляют средства управления записью дампа памяти. Например, в *bash* файлы дампов памяти по умолчанию не пишутся. Для того чтобы активировать запись дампов, следует выполнить команду:

```
ulimit -c unlimited
```

вероятно, в файле инициализации *.bashrc*. Можно указать максимальный размер файлов дампа, отличный от `unlimited`, но укороченные файлы могут оказаться бесполезными при отладке приложений.

Чтобы файл памяти оказался полезным, программа должна быть откомпилирована с включением отладочной информации, как описано в предыдущем разделе. В большинстве двоичных файлов вашей системы отладочной информации нет, поэтому файл с дампом памяти будет иметь ограниченную ценность.

Примером использования *gdb* с файлом дампа памяти послужит еще одна мифическая программа с названием *cross*. Как и *trymh* из предыдущего раздела, *cross* принимает на входе файл с изображением, производит над ним вычисления и выводит другой файл с изображением. Однако при исполнении *cross* мы встречаемся с ошибкой сегментации:

```
papaya$ cross < image30.pfm > image30.pbm
Segmentation fault (core dumped)
papaya$
```

Чтобы использовать *gdb* с файлом дампа памяти, нужно указать не только имя файла дампа памяти, но и имя исполняемого модуля, с которым он получен, поскольку файл дампа памяти не содержит всей необходимой для отладки информации:

```
papaya$ gdb cross core
GDB is free software and you are welcome to distribute copies of it
under certain conditions; type "show copying" to see the conditions.
There is absolutely no warranty for GDB; type "show warranty" for details.
GDB 4.8, Copyright 1993 Free Software Foundation, Inc...
Core was generated by `cross'.
Program terminated with signal 11, Segmentation fault.
#0 0x2494 in crossings (image=0xc7c8) at cross.c:31
31              if ((image[i][j] >= 0) &&
(gdb)
```

gdb сообщает, что этот файл дампа был создан, когда программа закончилась при получении сигнала с номером 11. *Сигнал* – это тип сообщения, которое посылается программе ядром, пользователем или самой программой. Сигналы обычно используются для завершения программы (и, возможно, вызывают создание файла с дампом памяти). Например, при вводе символа прерывания работающей программе посылается сигнал, который должен прервать ее исполнение.

В нашем случае ядро послало сигнал 11 процессу работающей программы *cross*, когда *cross* пыталась читать или писать в память, к которой у нее нет доступа. Этот сигнал привел к прекращению работы *cross* и созданию файла с дампом памяти. *gdb* сообщает, что недопустимое обращение к памяти произошло в строке 31 исходного файла *cross.c*:

```
(gdb) list
26          xmax = imGetWidth(image)`1;
27          ymax = imGetHeight(image)`1;
28
29          for (j=1; j<xmax; j++) {
30            for (i=1; i<ymax; i++) {
31              if ((image[i][j] >= 0) &&
32                  (image[i-1][j-1] < 0) ||
33                  (image[i-1][j] < 0) ||
34                  (image[i-1][j+1] < 0) ||
35                  (image[i][j-1] < 0) ||
(gdb)
```

Здесь обращают на себя внимание несколько вещей. Прежде всего, есть цикл по двум переменным `i` и `j`, вероятно, для производства вычислений с исходным изображением. В строке 31 осуществляется попытка обратиться к данным двумерного массива `image[i][j]`. Когда программа производит дамп при доступе к данным из массива, это обычно указывает на выход одного из индексов за пределы границ. Проверим их:

```
(gdb) print i
$1 = 1
(gdb) print j
$2 = 1194
(gdb) print xmax
$3 = 1551
(gdb) print ymax
$4 = 1194
(gdb)
```

Здесь видна проблема. Программа пыталась сослаться на элемент `image[1][1194]`, однако массив простирается только до `image[1550][1193]` (напомним, что массивы в языке C индексируются от `0` до `max-1`). Иными словами, мы пытались читать 1195-ю строку изображения, в котором только 1194 строки.

Если взглянуть на строки 29 и 30, то мы видим, в чем проблема: значения `xmax` и `ymax` переставлены местами. Переменная `j` должна изменяться от `1` до `ymax` (поскольку это индекс строк массива), а `i` должна изменяться от `1` до `xmax`. Исправление двух циклов в строках 29 и 30 устраняет проблему.

Допустим, наша программа создает аварийную ситуацию в функции, вызываемой из разных мест программы, и вы хотите определить, откуда была вызвана функция и что привело к аварии. Команда *backtrace* выводит стек вызовов программы в момент отказа. Если вам, как и автору, лень постоянно вводить с клавиатуры *backtrace*, могу к вашему удовольствию сообщить, что можно использовать сокращение *bt*.

Стек вызовов является списком функций, приведших в текущую функцию. Например, если программа начинается с функции `main`, вызывающей функцию `foo`, которая вызывает `bamf`, то стек вызовов будет выглядеть следующим образом:

```
(gdb) backtrace
#0 0x1384 in bamf () at goop.c:31
#1 0x4280 in foo () at goop.c:48
#2 0x218 in main () at goop.c:116
(gdb)
```

Каждая функция при своем вызове помещает некоторые данные в стек, например содержимое регистров, аргументы функции, локальные переменные и т. д. Каждой функции отводится определенное пространство в стеке, которое она может использовать. Участок памяти, выделенный в стеке для данной функции, называется *кадром стека* (*stack frame*), а стек вызовов является упорядоченным списком кадров стека.

В следующем примере мы видим дамп памяти программы анимации для X. Использование команды *backtrace* дает нам:

```
(gdb) backtrace
#0 0x602b4982 in _end ()
#1 0xbffff934 in _end ()
#2 0x13c6 in stream_drawimage (wgt=0x38330000, sn=4) at stream_display.c:94
#3 0x1497 in stream_refresh_all () at stream_display.c:116
#4 0x49c in control_update_all () at control_init.c:73
#5 0x224 in play_timeout (Cannot access memory at address 0x602b7676.
(gdb)
```

Это список кадров стека для процесса. Функции, вызванной последней, соответствует кадр 0. В нашем случае этой «функцией» является `_end`. Мы видим, что `play_timeout` вызвала `control_update_all`, которая вызвала `stream_refresh_all`, и т. д. Каким-то образом программа перешла на `_end`, где и завершилась с ошибкой.

Однако `_end` не является функцией – это просто метка, указывающая на конец сегмента данных процесса. Когда программа переходит на такой адрес, как `_end`, не являющийся реальной функцией, это значит, что по каким-то причинам процесс сбился, и стек вызовов поврежден (на языке хакеров это называется «прыжком в гиперпространство»). На практике сообщение `Cannot access memory at address 0x602b7676` является другим свидетельством того, что произошло нечто странное.

Мы видим, однако, что последней «реальной» вызванной функцией была `stream_drawimage`, и можно предположить, что источник проблем лежит в ней. Чтобы изучить состояние `stream_drawimage`, нужно выбрать ее кадр стека (кадр 2) с помощью команды *frame*:

```
(gdb) frame 2
#2 0x13c6 in stream_drawimage (wgt=0x38330000, sn=4) \
```

```
at stream_display.c:94
94        XCopyArea(mydisplay,streams[sn].frames[currentframe],\
XtWindow(wgt),
(gdb) list
91
92        printf("CopyArea frame %d, sn %d, wid %d\en",currentframe,sn,wgt);
93
94        XCopyArea(mydisplay,streams[sn].frames[currentframe],\
XtWindow(wgt),
95                  picGC,0,0,streams[sn].width,streams[sn].height,0,0);
(gdb)
```

Не зная больше ничего об этой программе, мы не можем сказать, в чем здесь дело, если только переменная `sn`, используемая в качестве индекса массива `streams`, не вышла за допустимые границы. Данные вывода *frame* показывают, что `stream_drawimage` была вызвана со значением переменной `sn`, равным `4`. (Параметры функций выводятся командой *backtrace*, а также при смене кадра.)

Переместимся в другой кадр, `stream_refresh_all`, чтобы посмотреть, как вызывалась `stream_display`. Для этого мы используем команду *up*, которая выбирает кадр стека, находящийся над текущим кадром:

```
(gdb) up
#3 0x1497 in stream_refresh_all () at stream_display.c:116
116        stream_drawimage(streams[i].drawbox,i);
(gdb) list
113     void stream_refresh_all(void) {
114       int i;
115       for (i=0; i<=numstreams; i++) {
116         stream_drawimage(streams[i].drawbox,i);
117
(gdb) print i
$2 = 4
(gdb) print numstreams
$3 = 4
(gdb)
```

Мы видим, что значение индексной переменной `i` изменяется от `0` до `numstreams`, и действительно `i` здесь равно `4` — второму параметру `stream_drawimage`. Однако `numstreams` тоже равно `4`. Что происходит?

Цикл `for` в строке 115 выглядит странно; он должен выглядеть так:

```
for (i=0; i<numstreams; i++) {
```

Ошибка заключается в использовании оператора сравнения `<=`. Индекс массива `streams` лежит в пределах от `0` до `numstreams-1`, а не от `0` до `numstreams`. Эта простая ошибка смещения на единицу заставила программу обезуметь.

Как видите, использование *gdb* с дампом памяти позволяет просматривать образ аварийно завершившейся программы и находить ошибки. Вы больше не будете удалять эти надоедливые файлы дампов?

Отладка работающей программы

gdb позволяет производить отладку программы, которая уже выполняется, предоставляя возможность прервать ее, изучить, а затем возобновить процесс обыч-

ного выполнения. Это весьма сходно с запуском программы из *gdb*, и вам потребуется изучить лишь несколько новых команд.

Команда *attach* присоединяет *gdb* к работающему процессу. Для того чтобы использовать *attach*, у вас должен быть доступ к исполняемому модулю, соответствующему этому процессу.

Например, если вы запустили программу *pgmseq* с идентификатором процесса 254, то можете запустить *gdb*:

```
papaya$ gdb pgmseq
```

а затем уже в *gdb* выполнить команду:

```
(gdb) attach 254
Attaching program `/home/loomer/mdw/pgmseq/pgmseq', pid 254
__select (nd=4, in=0xbffff96c, out=0xbffff94c, ex=0xbffff92c, tv=0x0)
at __select.c:22
__select.c:22: No such file or directory.
(gdb)
```

Ошибка `No such file or directory` выдается потому, что *gdb* не может найти исходный файл для `__select`. Это часто бывает с системными вызовами и библиотечными функциями и не должно вызывать тревогу.

Можно также запустить *gdb* командой:

```
papaya$ gdb pgmseq 254
```

После того как *gdb* присоединится к работающему процессу, он временно прекратит исполнение программы и передаст вам управление, позволив выполнять команды *gdb*. Вы можете установить точки останова или контрольные точки (командами *break* и *watch*, соответственно) и воспользоваться командой *continue*, чтобы продолжить выполнение до точки останова.

Команда *detach* отсоединяет *gdb* от текущего процесса. При необходимости можно подключиться к другому работающему процессу. Найдя ошибку, можно отсоединиться (*detach*) от текущего процесса, сделать изменения в исходном файле, перекомпилировать программу и использовать команду *file* для загрузки в *gdb* нового исполняемого модуля. Затем можно запустить новую версию программы и использовать команду *attach* для ее отладки. И все это – не покидая *gdb*!

Фактически *gdb* позволяет одновременно отлаживать три программы: одну, выполняемую непосредственно в *gdb*, вторую, трассируемую с помощью файла дампа памяти, и третью, работающую в качестве независимого процесса. Команда *target* позволяет выбрать программу, которую вы хотите отлаживать.

Изменение и изучение данных

Для изучения значений переменных в вашей программе можно использовать команды *print*, *x* и *ptype*. Команда *print* чаще всего используется для изучения данных. В качестве аргумента она принимает выражение на исходном языке (обычно C или C++) и возвращает его значение, например:

```
(gdb) print mydisplay
$10 = (struct _XDisplay *) 0x9c800
(gdb)
```

Эта команда выводит значение переменной `mydisplay` и ее тип. Поскольку данная переменная является указателем, ее содержимое можно изучить путем разыменования, как в языке C:

```
(gdb) print *mydisplay
$11 = {ext_data = 0x0, free_funcs = 0x99c20, fd = 5, lock = 0,
  proto_major_version = 11, proto_minor_version = 0,
  vendor = 0x9dff0 "XFree86", resource_base = 41943040,
  ...
  error_vec = 0x0, cms = {defaultCCCs = 0xa3d80 "",\
clientCmaps = 0x991a0 "",
  perVisualIntensityMaps = 0x0}, conn_checker = 0, im_filters = 0x0}
(gdb)
```

`mydisplay` — это обширная структура, используемая программами для X. Для удобства чтения мы сократили вывод.

print может выводить значения почти любых выражений, в том числе вызовов функций C, которые она выполняет «на лету» в контексте работающей программы:

```
(gdb) print getpid()
$11 = 138
(gdb)
```

Конечно, так можно вызывать не все функции, а только те, которые скомпонованы с выполняемой программой. Если попытаться вызвать функцию, не скомпонованную с программой, *gdb* пожалуется, что такого символа нет в текущем контексте.

В качестве аргументов *print* могут использоваться более сложные выражения, включая присваивание значений переменным, например:

```
(gdb) print mydisplay->vendor = "Linux"
$19 = 0x9de70 "Linux"
(gdb)
```

присваивает элементу `vendor` структуры `mydisplay` значение `"Linux"` вместо `"XFree86"` (бесполезная модификация, интересная лишь в качестве примера). Таким способом можно интерактивно изменять данные в работающей программе для того, чтобы исправить ошибки или проверить необычные ситуации.

Обратите внимание: после каждой команды *print* выведенное значение присваивается одному из регистров *gdb*, являющихся внутренними переменными *gdb*, используемыми для удобства. Например, чтобы вспомнить значение `mydisplay` в предыдущем примере, нужно лишь вывести значение `$10`:

```
(gdb) print $10
$21 = (struct _XDisplay *) 0x9c800
(gdb)
```

В команде *print* можно использовать такие выражения, как приведение типа; годится почти все.

Команда *ptype* дает подробные (и часто многословные) сведения о типе переменной либо об определении структуры (*struct*) или типа (*typedef*). Для того чтобы

получить полное определение структуры `struct _Xdisplay`, используемой переменной `mydisplay`, выполним команду:

```
(gdb) ptype mydisplay
type = struct _XDisplay {
    struct _XExtData *ext_data;
    struct _XFreeFuncs *free_funcs;
    int fd;
    int lock;
    int proto_major_version;
    ...
    struct _XIMFilter *im_filters;
} *
(gdb)
```

Если вы хотите более фундаментально изучить память, не ограничиваясь определенными типами, можете выполнить команду *x*. *x* принимает в качестве аргумента адрес памяти. Если передать ей переменную, то в качестве адреса используется значение этой переменной. В виде необязательного аргумента можно передать команде *x* количество и тип данных. Количество – это число выводимых объектов данного типа. Например, `x/100x 0x4200` выводит 100 байт данных, представленных в шестнадцатеричном формате, начиная с адреса `0x4200`. Описание различных форматов вывода можно получить командой *help x*.

Чтобы выяснить значение `mydisplay->vendor`, можно выполнить команду:

```
(gdb) x mydisplay->vendor
0x9de70 <_end+35376>:  76 'L'
(gdb) x/6c mydisplay->vendor
0x9de70 <_end+35376>:  76 'L' 105 'i' 110 'n' 117 'u' 120 'x' 0 '\000'
(gdb) x/s mydisplay->vendor
0x9de70 <_end+35376>:    "Linux"
(gdb)
```

Первое поле каждой строки показывает абсолютный адрес данных. Во втором поле адрес представлен как некоторый символ (в данном случае `_end`) плюс смещение в байтах. В остальных полях дается фактическое содержимое памяти по этому адресу: сначала как десятичное число, а затем как символ ASCII. Как уже говорилось, можно потребовать от *x* вывода данных в других форматах.

Получение информации

При помощи команды *info* можно получить данные о состоянии отлаживаемой программы. В *info* есть много подкоманд; чтобы вывести их список, выполните команду *help info*. Например, *info program* выводит статус исполнения программы:

```
(gdb) info program
Using the running image of child process 138.
Program stopped at 0x9e.
It stopped at breakpoint 1.
(gdb)
```

Другой полезной командой является *info locals*, которая выводит имена и значения всех локальных переменных в текущей функции:

```
(gdb) info locals
inimage = (struct {...} *) 0x2000
outimage = (struct {...} *) 0x8000
(gdb)
```

Это довольно беглое описание переменных. Команды *print* или *x* описывают их подробнее.

Аналогично *info variables* выводит список всех известных переменных в программе, определяемых исходным файлом. Обратите внимание, что многие выводимые переменные относятся к источникам, находящимся вне вашей фактической программы, например в библиотечном коде. Значения этих переменных не выводятся, поскольку список составляется более или менее непосредственно из таблицы символов выполняемого модуля. Фактически *gdb* доступны только локальные переменные в текущем кадре стека и глобальные (статические) переменные. Команда *info address* дает вам сведения о том, где хранится некоторая переменная, например:

```
(gdb) info address inimage
Symbol "inimage" is a local variable at frame offset -20.
(gdb)
```

Значение `frame offset` говорит о том, что переменная `inimage` хранится на 20 байт ниже вершины кадра стека.

При помощи команды *info frame* можно получить данные о текущем кадре стека, например:

```
(gdb) info frame
Stack level 0, frame at 0xbffffaa8:
 eip = 0x9e in main (main.c:44); saved eip 0x34
 source language c.
 Arglist at 0xbffffaa8, args: argc=1, argv=0xbffffabc
 Locals at 0xbffffaa8, Previous frame's sp is 0x0

 Saved registers:
  ebx at 0xbffffaa0, ebp at 0xbffffaa8, esi at 0xbffffaa4, eip at\
0xbffffaac
(gdb)
```

Такого рода данные полезны при отладке на уровне языка ассемблера при помощи команд *disass*, *nexti* и *stepi* (подробности в разделе «Отладка на уровне инструкций» далее в этой главе).

Разные функции

Мы лишь поверхностно коснулись того, что может делать *gdb*. Это удивительная и очень мощная программа. Мы познакомили вас только с наиболее часто используемыми командами. В этом разделе будут рассмотрены другие функции *gdb*.

Если вы хотите больше узнать о *gdb*, рекомендуем почитать страницу справочного руководства по *gdb* и руководство Фонда свободного программного обеспечения. Руководство имеется также и в виде Info-файла. (Файлы Info можно читать в Emacs или с помощью программы *info*, о чем рассказывалось в разделе «Учебное руководство и оперативная подсказка» главы 19.)

Точки останова и контрольные точки

Как мы и обещали, приведем еще примеры использования точек останова и контрольных точек. Точки останова устанавливаются командой *break*; аналогично контрольные точки устанавливаются командой *watch*. Единственное различие между теми и другими состоит в том, что точки останова вызывают прерывание выполнения в определенном месте программы, например на некоторой строке программного кода, а контрольные точки срабатывают, когда некоторое выражение становится истинным, независимо от места в программе, где это происходит. Несмотря на всю их мощь, контрольные точки могут быть крайне неэффективными: при каждом изменении состояния программы пересчитываются все контрольные точки.

Когда срабатывает точка останова или контрольная точка, *gdb* приостанавливает исполнение программы и передает вам управление. Точки останова и контрольные точки позволяют выполнять программу (используя команды *run* и *continue*) и останавливаться только в определенных ситуациях, избавляя от необходимости ручного прохода программы с помощью многочисленных команд *next* и *step*.

Существует много способов установки точки останова в программе. Можно указать номер строки, например в команде *break 20*, или задать конкретную функцию, например *break stream_unload*. Можно также задать номер строки в другом исходном файле, например *break foo.c:38*. Полный синтаксис можно получить по команде *help break*.

Точки останова могут быть условными. Это означает, что точка останова срабатывает только при истинности некоторого выражения. Например, команда

```
break 184 if (status = = 0)
```

устанавливает условную точку останова в строке 184 текущего исходного файла, которая срабатывает только тогда, когда переменная `status` равна нулю. Переменная `status` должна быть либо глобальной, либо локальной в текущем кадре стека. Выражение может быть любым допустимым в исходном языке выражением, которое понятно *gdb*, подобно выражениям, используемым командой *print*. Изменить условие точки останова можно с помощью команды *condition*.

Команда *info break* выводит список всех точек останова и контрольных точек вместе с их статусом. Это позволяет удалять или отключать точки останова, используя команды *clear*, *delete* или *disable*. Отключенная точка останова просто бездействует до тех пор, пока не будет реактивирована командой *enable*. Напротив, при удалении точка останова удаляется из списка навсегда. Можно также потребовать, чтобы точка останова срабатывала однократно, после чего отключалась.

Для установки контрольной точки выполните команду *watch*:

```
watch (numticks < 1024 && incoming != clear)
```

Контрольная точка может иметь условие, являющееся любым допустимым выражением исходного кода, таким же, как для точек останова.

Отладка на уровне инструкций

С помощью *gdb* можно производить отладку на уровне инструкций процессора, что позволяет самым пристальным образом рассматривать содержимое вашей программы. Однако для понимания того, что вы видите, требуется не только

знание архитектуры процессора и языка ассемблера, но и некоторое понимание того, как операционная система организует адресное пространство процесса. Например, полезно понимать соглашения, используемые при создании кадров стека, вызове функций, передаче параметров и получении возвращаемых значений и т. д. Любая книга по программированию в защищенном режиме 80386/80486 подробно ознакомит вас с этими вопросами. Но имейте в виду, что программирование в защищенном режиме этого процессора совершенно отличается от программирования в реальном режиме (используемом в мире MS-DOS). Удостоверьтесь, что вы читаете именно о программировании в защищенном режиме 386, иначе будете периодически путаться.

Главными командами *gdb*, используемыми при отладке на уровне инструкций, являются *nexti*, *stepi* и *disass*. *nexti* эквивалентна команде *next*, но осуществляет переход к следующей инструкции, а не к следующей строке исходного кода. Аналогично *stepi* является аналогом *step* на уровне инструкций.

Команда *disass* дизассемблирует переданный ей диапазон адресов. Этот диапазон может быть задан точным адресом или как имя функции. Например, для дизассемблирования функции `play_timeout` выполните команду:

```
(gdb) disass play_timeout
Dump of assembler code for function play_timeout:
to 0x2ac:
0x21c <play_timeout>:           pushl  %ebp
0x21d <play_timeout+1>:         movl   %esp,%ebp
0x21f <play_timeout+3>:         call   0x494 <control_update_all>
0x224 <play_timeout+8>:         movl   0x952f4,%eax
0x229 <play_timeout+13>:        decl   %eax
0x22a <play_timeout+14>:        cmpl   %eax,0x9530c
0x230 <play_timeout+20>:        jne    0x24c <play_timeout+48>
0x232 <play_timeout+22>:        jmp    0x29c <play_timeout+128>
0x234 <play_timeout+24>:        nop
0x235 <play_timeout+25>:        nop
...
0x2a8 <play_timeout+140>:       addb   %al,(%eax)
0x2aa <play_timeout+142>:       addb   %al,(%eax)
(gdb)
```

Это эквивалентно использованию команды *disass 0x21c* (где `0x21c` – точный адрес начала функции `play_timeout`).

Можно передать команде *disass* необязательный второй аргумент, который будет использован как адрес, по которому дизассемблирование прекратится. Команда *disass 0x21c 0x232* выведет только первые семь строк листинга ассемблера предыдущего примера (инструкция, начинающаяся с адреса `0x232`, выведена не будет).

При частом использовании команд *nexti* и *stepi* может пригодиться команда:

```
display/i $pc
```

Она вызывает вывод текущей команды после каждой выполненной *nexti* или *stepi*. *display* задает переменные, которые нужно просмотреть, или команды, которые нужно выполнить после каждой команды пошагового выполнения. `$pc` является внутренним регистром *gdb*, соответствующим программному счетчику процессора, указывающему на очередную инструкцию.

Использование Emacs с gdb

(X)Emacs (описанный в главе 19) предоставляет режим отладки, который позволяет запустить *gdb* или другой отладчик в интегрированной среде трассировки программ, предоставляемой Emacs. Эта так называемая библиотека Grand Unified Debugger (библиотека большого унифицированного отладчика) – очень мощная библиотека, позволяющая редактировать и отлаживать программы целиком внутри Emacs.

Для запуска *gdb* под Emacs выполните команду Emacs `M-x gdb` и передайте ей в качестве аргумента имя исполняемого модуля. Для *gdb* будет создан буфер, что аналогично отдельному использованию *gdb*. После этого при желании вы можете использовать команду *core-file* для загрузки файла дампа памяти или команду *attach* для подключения к исполняемому процессу.

При переходе к новому кадру стека (например, при первой остановке в точке останова) *gdb* открывает отдельное окно, в котором выводится исходный код, соответствующий текущему кадру стека. Этот буфер может использоваться для редактирования исходного кода так же, как обычно делается в Emacs, при этом текущая строка исходного текста отмечена стрелкой (`=>`). Это позволяет видеть исходный текст в одном окне и выполнять команды *gdb* в другом.

В окне отладки можно использовать несколько специальных последовательностей символов. Однако они достаточно длинные, и их использование может показаться не более удобным, чем непосредственный ввод команд *gdb*. В число наиболее часто употребляемых команд входят:

`C-x C-a C-s`

Эквивалент команды *gdb step*, соответствующим образом обновляющий окно с исходным кодом.

`C-x C-a C-i`

Эквивалент команды *stepi*.

`C-x C-a C-n`

Эквивалент команды *next*.

`C-x C-a C-r`

Эквивалент команды *continue*.

`C-x C-a <`

Эквивалент команды *up*.

`C-x C-a >`

Эквивалент команды *down*.

Если вы вводите команды в обычном стиле, то можно использовать комбинацию `M-p` для перемещения назад по буферу введенных ранее команд и `M-n` для перемещения вперед. Можно также перемещаться по буферу с использованием команд Emacs для поиска, перемещения курсора и т. д. В целом использование *gdb* в Emacs более удобно, чем из командной оболочки.

Кроме того, можно редактировать исходный текст в буфере с исходным текстом *gdb*. Префиксная стрелка при сохранении текста записываться не будет.

Emacs очень легко настраивается, и многие расширения для этого интерфейса с *gdb* вы можете написать сами. В Emacs можно определить комбинации клавиш для часто используемых в *gdb* команд или изменить характер поведения окна с исходным текстом. (Например, можно каким-то образом выделить все точки останова или определить клавиши для отключения или удаления точек останова.)

Полезные утилиты для C-программистов

Наряду с языками и компиляторами существует изобилие программных средств, в том числе библиотеки, конструкторы интерфейсов, отладчики и прочие средства поддержки процесса программирования. В этом разделе мы поговорим о некоторых наиболее интересных «бантиках и рюшечках» среди этих средств, чтобы вы знали об имеющихся инструментах.

Отладчики

В Linux существует несколько интерактивных отладчиков. Стандартом де-факто является *gdb*, о котором мы только что подробно рассказали.

Кроме *gdb* есть несколько других отладчиков со схожими характеристиками. DDD (Data Display Debugger) является разновидностью *gdb* для работы в среде X Window System[1], имеет интерфейс, напоминающий отладчик *xdbx*, используемый в других UNIX-системах. В окне отладчика DDD имеется несколько панелей. Одна из них предоставляет интерфейс, напоминающий обычный текстовый интерфейс *gdb*, что позволяет вручную вводить команды взаимодействия с системой. В другой панели автоматически отображается содержимое файла с исходными текстами вместе с меткой, отмечающей текущую строку. В панели с исходными текстами можно устанавливать и выбирать точки останова, просматривать исходный текст и т. д. простым вводом команд *gdb*. Кроме того, окно DDD содержит несколько кнопок, обеспечивающих быстрый доступ к часто используемым командам, таким как *next*, *step* и тому подобным. Благодаря наличию этих кнопок в удобном графическом интерфейсе отладку программ можно производить с использованием не только клавиатуры, но и мыши. Наконец, отладчик DDD имеет очень удобный режим, в котором предоставляется возможность исследования структур данных неизвестных программ.

Среда разработки KDevelop поставляется со своим собственным, очень удобным графическим интерфейсом к *dbg*. Он полностью интегрирован в рабочий стол KDE. Подробнее о KDevelop мы поговорим в конце главы.

Средства профилирования

Существует несколько утилит, позволяющих контролировать и измерять производительность программ. Эти средства помогают выявлять узкие места в программном коде, то есть те места, которым недостает производительности. Они

[1] DDD не автономный отладчик, а лишь удобная для работы интерактивная «обертка» (front-end) для *gdb*, который и является внутренним «движком» DDD (без *gdb* работать DDD не будет). Поэтому неудивительно, что в DDD можно использовать все богатство возможностей *gdb*. – *Примеч. науч. ред.*

позволяют также получить схему структуры вызовов в вашей программе с указанием того, какие функции вызываются, откуда и как часто. (Все, что вы хотели знать о вашей программе, но боялись спросить.)

gprof является утилитой профилирования, предоставляющей детальный листинг статистики выполнения вашей программы, включая частоту вызова всех функций и мест, откуда они вызывались, суммарный объем времени, затраченный на выполнение каждой функции, и т. д.[1]

Чтобы использовать *gprof* вместе с программой, нужно компилировать ее в *gcc* с ключом *-pg*. Благодаря этому в объектный файл добавляются данные для профилирования, и он компонуется со стандартными библиотеками, в которых есть данные для профилирования.

Скомпилировав программу с ключом *-pg*, просто запустите ее. Если она нормально завершит работу, то в рабочий каталог программы будет записан файл *gmon.out*. В нем содержатся данные по профилированию этого прогона, на основе которых *gprof* выведет таблицу статистики.

Для примера возьмем программу с именем *getstat*, собирающую статистические данные о графическом файле. Сначала мы скомпилируем ее с ключом *-pg*, а затем запустим:

```
papaya$ getstat image11.pgm > stats.dat
papaya$ ls -l gmon.out
-rw-------   1 mdw      mdw 54448 Feb 5 17:00 gmon.out
papaya$
```

Данные профилирования действительно были записаны в файл *gmon.out*.

Для изучения данных профилирования запустим *gprof*, передав ей имя исполняемого модуля и файла профилирования *gmon.out*:

```
papaya$ gprof getstat gmon.out
```

Если имя файла профилирования не задано, *gprof* предполагает имя *gmon.out*. Она также предполагает, что имя выполняемого модуля – *a.out*, если и оно не указано.

Вывод *gprof* довольно многословен, поэтому вы можете перенаправить его в файл или программу постраничного вывода. Вывод состоит из двух частей. Первая часть является «плоским профилем» с одной строкой для каждой функции, в которой указан процент времени работы, потраченного на эту функцию, время (в секундах) ее выполнения, количество вызовов и другие данные, например:

```
Each sample counts as 0.01 seconds.
 %   cumulative   self              self     total
time   seconds   seconds    calls  ms/call  ms/call  name
45.11     27.49    27.49       41   670.51   903.13  GetComponent
```

[1] Все средства профилирования нужны с единственной, редко возникающей (но другими способами не достигаемой) потребностью – алгоритмической оптимизацией кода (ручной, в отличие от формальной оптимизации, выполняемой *gcc*). Известно «правило 5%»: 95% времени выполнения любой программы забирают 5% ее текстуального кода; для выявления этих «узких мест» и оказываются незаменимыми программы профилирования. – *Примеч. науч. ред.*

```
 16.25      37.40     9.91                                    mcount
 10.72      43.93     6.54  1811863     0.00     0.00  Push
 10.33      50.23     6.30  1811863     0.00     0.00  Pop
  5.87      53.81     3.58       40    89.50   247.06  stackstats
  4.92      56.81     3.00  1811863     0.00     0.00  TrimNeighbors
```

Если какие-либо поля оказываются пустыми, значит, *gprof* не смогла определить дополнительные данные относительно этой функции. Обычно это связано с тем, что некоторые части кода компилировались без ключа *-pg*. Например, при вызове функций из нестандартных библиотек, компилировавшихся без ключа *-pg*, *gprof* сможет собрать об этих функциях не много данных. В приведенном примере функция *mcount*, вероятно, компилировалась без включения профилирования.

Как можно видеть, 45,11% общего времени прогона было потрачено на функцию `GetComponent`, то есть в сумме 27,49 секунды. Но связано ли это с неэффективностью самой `GetComponent`[1] или с тем, что `GetComponent` вызывает много других медлительных функций? Во время выполнения многократно вызывались функции `Push` и `Pop`. Не они ли виновники?

Здесь нам может помочь вторая часть отчета *gprof*. В ней приведен подробный «граф вызовов», описывающий вызовы одних функций другими и частоту этих вызовов. Например:

```
index % time     self  children    called     name
                                                <spontaneous>
[1]     92.7     0.00    47.30                 start [1]
                 0.01    47.29       1/1           main [2]
                 0.00    0.00        1/2           on_exit [53]
                 0.00    0.00        1/1           exit [172]
```

Первая колонка графа вызовов является индексом – уникальным номером, присвоенным каждой функции, позволяющим отыскивать ее в графе. Первая функция, `start`, неявно вызывается при запуске программы. `start` потребовала 92,7% общего времени работы (47,30 секунды), включая ее потомков, но для самой себя потребовала очень мало времени. Это связано с тем, что `start` является родительской для всех остальных функций программы, включая `main`. Именно поэтому `start` вместе со своими потомками заняла такой процент времени.

Граф вызовов обычно отображает как потомков, так и родителей каждой функции графа. Мы видим, что `start` вызывала функции `main`, `on_exit` и `exit` (перечисленные под строкой для `start`). Однако у нее нет родителей (обычно перечисляемых над строкой `start`). Вместо этого мы видим зловещее слово `<spontaneous>`. Это означает, что `gprof` не смогла определить родителя `start`. Скорее всего, это вызвано тем, что `start` из самой программы не вызывалась, а была запущена операционной системой.

Перескакивая вниз к `GetComponent`, то есть к подозреваемой функции, мы видим следующее:

```
index % time     self  children    called     name
                 0.67    0.23        1/41          GetFirstComponent [12]
```

[1] Что всегда возможно, когда речь заходит о программном коде, автором которого вы являетесь!

```
                26.82    9.30      40/41          GetNextComponent [5]
[4]     72.6    27.49    9.54      41         GetComponent [4]
                 6.54    0.00 1811863/1811863     Push [7]
                 3.00    0.00 1811863/1811863     TrimNeighbors [9]
                 0.00    0.00       1/1           InitStack [54]
```

Родительскими функциями для `GetComponent` были `GetFirstComponent` и `GetNextComponent`, а дочерними – `Push`, `TrimNeighbors` и `InitStack`. Мы видим, что `GetComponent` вызывалась 41 раз (один раз из `GetFirstComponent` и 40 раз из `GetNextComponent`). В выводе *gprof* содержатся примечания, описывающие отчет более подробно.

Самой функции `GetComponent` требуется для выполнения 27,49 секунды, и лишь 9,54 секунды потрачено на выполнение дочерних функций `GetComponent` (включая многочисленные вызовы `Push` и `TrimNeighbors`!). По всей видимости, `GetComponent` и, возможно, ее родительская функция `GetNextComponent` нуждаются в доработке; часто вызываемая функция `Push` не является единственной причиной проблемы.

gprof отслеживает также рекурсивные вызовы и «циклы» вызываемых функций и указывает время, потраченное на каждый вызов. Конечно, для эффективного использования *gprof* требуется, чтобы весь профилируемый код компилировался с параметром *-pg*. Требуется также знать программу, которую вы пытаетесь профилировать. *gprof* может только сообщить о том, что делается, а оптимизация неэффективного кода остается за программистом.

Следует отметить, что использование *gprof* может не дать осмысленных результатов в случае профилирования программы, содержащей малое количество вызовов и очень быстро выполняющейся. Единицы измерения временных затрат обычно достаточно грубы (возможно, сотые доли секунды), и если многие функции вашей программы выполняются быстрее, *gprof* не сможет выявить различие во времени их выполнения, округляя результаты до сотых долей секунды. Для того чтобы получить правильную информацию профилирования в таком случае, можно выполнить программу в необычных условиях, например, дав ей на обработку необычно большой объем данных, как в предыдущем примере.

Если *gprof* превосходит ваши потребности, то существует программа *calls*[1], выводящая дерево вызовов функций в вашем исходном коде C. Она может оказаться полезной при создании указателя всех вызываемых функций или при создании иерархического отчета высокого уровня о структуре программы.

Использование *calls* отличается простотой: вы сообщаете имена исходных файлов, для которых должна быть составлена карта, и выводится дерево вызовов функций. Например:

```
papaya$ calls scan.c
    1   level1 [scan.c]
    2           getid [scan.c]
    3                   getc
    4                   eatwhite [scan.c]
    5                           getc
    6                           ungetc
```

[1] Присутствует далеко не во всех дистрибутивах, но, как и любые другие средства, может быть установлена дополнительно. – *Примеч. науч. ред.*

```
 7                  strcmp
 8          eatwhite [see line 4]
 9          balance [scan.c]
10                  eatwhite [see line 4]
```

По умолчанию *calls* перечисляет только по одному экземпляру каждой вызываемой функции на каждом уровне дерева (поэтому, если `printf` вызывается в данной функции пять раз, она будет перечислена лишь однажды). Ключ *-a* позволяет вывести все случаи вызовов. *calls* имеет и другие параметры, перечень которых можно получить при помощи *calls -h*.

Использование strace

strace выводит на экран системные вызовы, осуществляемые работающей программой. Эта утилита может оказаться исключительно полезной при мониторинге работы программы в реальном времени, хотя при этом потребуется некоторое знание программирования на уровне системных вызовов. Например, когда в программе используется библиотечная функция `printf`, *strace* выводит данные только о выполнении лежащего в ее основе системного вызова `write`. Кроме того, вывод *strace* может оказаться очень пространным: в программе может выполняться большое количество системных вызовов, о которых программист и не подозревает. Однако *strace* полезна для быстрого определения причины отказа программы или иного необычного поведения.

Возьмем программу «Hello, World!», приведенную в начале главы. Запуск *strace* с выполняемым модулем *hello* дает следующий результат:

```
papaya$ strace hello
execve("./hello", ["hello"], [/* 49 vars */]) = 0
mmap(0, 4096, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS, \
-1, 0) = 0x40007000
mprotect(0x40000000, 20881, PROT_READ|PROT_WRITE|PROT_EXEC) = 0
mprotect(0x8048000, 4922, PROT_READ|PROT_WRITE|PROT_EXEC) = 0
stat("/etc/ld.so.cache", {st_mode=S_IFREG|0644, st_size=18612,\
...}) = 0
open("/etc/ld.so.cache", O_RDONLY)      = 3
mmap(0, 18612, PROT_READ, MAP_SHARED, 3, 0) = 0x40008000
close(3)                                = 0
stat("/etc/ld.so.preload", 0xbffff52c)  = -1 ENOENT (No\
 such file or directory)
open("/usr/local/KDE/lib/libc.so.5", O_RDONLY) = -1 ENOENT (No\
 such file or directory)
open("/usr/local/qt/lib/libc.so.5", O_RDONLY) = -1 ENOENT (No\
 such file or directory)
open("/lib/libc.so.5", O_RDONLY)        = 3
read(3, "ELF\1\1\1\0\0\0\0\0\0\0\0\0\3"..., 4096) = 4096
mmap(0, 770048, PROT_NONE, MAP_PRIVATE|MAP_ANONYMOUS, -1, 0) = \
0x4000d000
mmap(0x4000d000, 538959, PROT_READ|PROT_EXEC, MAP_PRIVATE|MAP_\
FIXED, 3, 0) = 0x4000d000
mmap(0x40091000, 21564, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_\
FIXED, 3, 0x83000) = 0x40091000
mmap(0x40097000, 204584, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_\
FIXED|MAP_ANONYMOUS, -1, 0) = 0x40097000
```

```
close(3)                                = 0
mprotect(0x4000d000, 538959, PROT_READ|PROT_WRITE|PROT_EXEC) = 0
munmap(0x40008000, 18612)               = 0
mprotect(0x8048000, 4922, PROT_READ|PROT_EXEC) = 0
mprotect(0x4000d000, 538959, PROT_READ|PROT_EXEC) = 0
mprotect(0x40000000, 20881, PROT_READ|PROT_EXEC) = 0
personality(PER_LINUX)                  = 0
geteuid()                               = 501
getuid()                                = 501
getgid()                                = 100
getegid()                               = 100
fstat(1, {st_mode=S_IFCHR|0666, st_rdev=makedev(3, 10), ...}) = 0
mmap(0, 4096, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS,\
 -1, 0) = 0x40008000
ioctl(1, TCGETS, {B9600 opost isig icanon echo ...}) = 0
write(1, "Hello World!\n", 13Hello World!
)          = 13
_exit(0)                                = ?
papaya$
```

Это значительно больше, чем можно было ожидать от простой программы. Рассмотрим этот вывод и кратко объясним, что происходит.

Первый вызов `execve` запускает саму программу. Все вызовы `mmap`, `mprotect` и `munmap` исходят от системы управления памятью ядра и не очень нам здесь интересны. В трех последующих вызовах `open` загрузчик ищет библиотеку С и находит ее с третьей попытки. После этого считывается заголовок библиотеки, и библиотека отображается в память. После нескольких дополнительных операций управления памятью и вызовов `geteuid`, `getuid`, `getgid` и `getegid`, которые извлекают привилегии процесса, следует вызов `ioctl`. Обращение к `ioctl` является результатом библиотечного вызова `tcgetattr`, который программа использует для извлечения атрибутов терминала, прежде чем пытаться выводить на терминал. И, наконец, вызов `write` выводит на терминал наше дружелюбное сообщение, а `exit` завершает программу.

strace отправляет результаты своей работы на стандартное устройство вывода ошибок, поэтому вы можете переназначить ее в файл отдельно от того, что выводит программа (обычно отправляется на стандартное устройство вывода). Как видите, *strace* сообщает не только имена системных вызовов, но и их параметры (в виде хорошо известных имен констант, а не просто чисел, если это возможно) и возвращаемые значения.

Не менее удобным является пакет *ltrace*. Это трассировщик библиотечных вызовов, который отслеживает не только вызовы функций в ядре, но и все обращения к библиотечным функциям. Некоторые дистрибутивы уже включают в себя эту утилиту; пользователи других дистрибутивов могут загрузить последнюю версию исходных текстов с сайта *ftp://ftp.debian.org/debian/dists/unstable/main/source/utils/*.

Использование библиотеки Valgrind

Библиотека Valgrind заменяет многочисленные функции распределения памяти, такие как `malloc`, `realloc` и `free`, используемые программами, написанными

на языке C, но поддерживает также программы C++. Она обеспечивает более интеллектуальные процедуры выделения памяти и средства, позволяющие определять несанкционированный доступ к памяти, а также частые ошибки, такие как попытки повторного освобождения блока памяти. Valgrind выводит подробные сообщения об ошибках, когда ваша программа делает опасные попытки доступа к памяти, помогая перехватить ошибки сегментации до того, как они произойдут. Она может также выявлять утечки памяти, например места в программном коде, где `malloc` выделяет новую память без последующего освобождения функцией `free` после ее использования.

Valgrind не просто заменяет `malloc` и другие функции. Она вставляет в программу код, проверяющий все операции чтения и записи в память. Эта библиотека обладает большой надежностью, и поэтому выполняется существенно медленнее, чем обычные функции `malloc`. Valgrind предназначена для использования во время разработки и тестирования программ. После устранения всех потенциальных ошибок обращения к памяти вы можете запустить свою программу без Valgrind.

Рассмотрим программу, которая выделяет некоторый объем памяти и пытается делать с ней нечто невообразимое:

```
#include <malloc.h>
int main() {
    char *thememory, ch;

    thememory=(char *)malloc(10*sizeof(char));

    ch=thememory[1];    /* Попытка чтения неинициализированной памяти */
    thememory[12]=' ';  /* Попытка записи после блока выделенной памяти */
    ch=thememory[-2];   /* Попытка чтения перед блоком выделенной памяти */
}
```

Чтобы найти эти ошибки, скомпилируем программу для отладки и запустим, введя предварительно в командной строке *valgrind*:

```
owl$ gcc -g -o nasty nasty.c
owl$ valgrind nasty
=  =18037=  = valgrind-20020319, a memory error detector for x86 GNU/Linux.
=  =18037=  = Copyright (C) 2000-2002, and GNU GPL'd, by Julian Seward.
=  =18037=  = For more details, rerun with: -v
=  =18037=  =
=  =18037=  = Invalid write of size 1
=  =18037=  =  at 0x8048487: main (nasty.c:8)
=  =18037=  =  by 0x402D67EE: __libc_start_main (in /lib/libc.so.6)
=  =18037=  =  by 0x8048381: __libc_start_main@@GLIBC_2.0 (in /home/kalle/tmp/nasty)
=  =18037=  =  by <bogus frame pointer> ???
=  =18037=  =  Address 0x41B2A030 is 2 bytes after a block of size 10 alloc'd
=  =18037=  =  at 0x40065CFB: malloc (vg_clientmalloc.c:618)
=  =18037=  =  by 0x8048470: main (nasty.c:5)
=  =18037=  =  by 0x402D67EE: __libc_start_main (in /lib/libc.so.6)
=  =18037=  =  by 0x8048381: __libc_start_main@@GLIBC_2.0 (in /home/kalle/tmp/nasty)
=  =18037=  =
=  =18037=  =  Invalid read of size 1
=  =18037=  =  at 0x804848D: main (nasty.c:9)
=  =18037=  =  by 0x402D67EE: __libc_start_main (in /lib/libc.so.6)
=  =18037=  =  by 0x8048381: __libc_start_main@@GLIBC_2.0 (in /home/kalle/tmp/nasty)
```

```
=  =18037=  =  by <bogus frame pointer> ???
=  =18037=  =  Address 0x41B2A022 is 2 bytes before a block of size 10 alloc'd
=  =18037=  =  at 0x40065CFB: malloc (vg_clientmalloc.c:618)
=  =18037=  =  by 0x8048470: main (nasty.c:5)
=  =18037=  =  by 0x402D67EE: __libc_start_main (in /lib/libc.so.6)
=  =18037=  =  by 0x8048381: __libc_start_main@@GLIBC_2.0 (in /home/kalle/tmp/nasty)
=  =18037=  =
=  =18037=  =  ERROR SUMMARY: 2 errors from 2 contexts (suppressed: 0 from 0)
=  =18037=  =  malloc/free: in use at exit: 10 bytes in 1 blocks.
=  =18037=  =  malloc/free: 1 allocs, 0 frees, 10 bytes allocated.
=  =18037=  =  For a detailed leak analysis, rerun with: --leak-check=yes
=  =18037=  =  For counts of detected errors, rerun with: -v
```

Число в начале каждой строки указывает ID процесса. Если процесс порождает другие процессы, то и они будут выполняться под управлением Valgrind.

В случае любого нарушения правил работы с памятью Valgrind сообщает об ошибке и выводит данные о том, что произошло. Действительные сообщения Valgrind об ошибках включают в себя данные о том, в каком месте выполняется программа и где был размещен блок памяти. При желании вы можете получить от Valgrind более подробные сведения, а если дополнительно использовать отладчик, например *gdb*, можно легко локализовать ошибки.

Вы можете задать вопрос, почему операция чтения в строке 7, обращающаяся к неинициализированному участку памяти, не заставила Valgrind выдать сообщение об ошибке. Это связано с тем, что Valgrind не выводит сообщений об ошибках, когда передается инициализированная память, но продолжает следить за ней. Как только вы воспользуетесь значением (например, при передаче функции операционной системы или при его обработке), вы получите ожидаемое сообщение об ошибке.

Valgrind предоставляет также сборщик и детектор мусора, которые можно вызывать из вашей программы. Детектор мусора сообщает об утечках памяти, то есть о тех местах, где `malloc` выделила блок памяти, но до выхода не была выполнена соответствующая функция `free`. Сборщик мусора просматривает «кучу» и очищает ее от результатов этих утечек. Ниже приводится пример вывода:

```
owl$ valgrind -leak-check=yes -show-reachable=yes nasty
...
=  =18081=  =  ERROR SUMMARY: 2 errors from 2 contexts (suppressed: 0 from 0)
=  =18081=  =  malloc/free: in use at exit: 10 bytes in 1 blocks.
=  =18081=  =  malloc/free: 1 allocs, 0 frees, 10 bytes allocated.
=  =18081=  =  For counts of detected errors, rerun with: -v
=  =18081=  =  searching for pointers to 1 not-freed blocks.
=  =18081=  =  checked 4029376 bytes.
=  =18081=  =
=  =18081=  =  definitely lost: 0 bytes in 0 blocks.
=  =18081=  =  possibly lost: 0 bytes in 0 blocks.
=  =18081=  =  still reachable: 10 bytes in 1 blocks.
=  =18081=  =
=  =18081=  =  10 bytes in 1 blocks are still reachable in loss record 1 of 1
=  =18081=  =  at 0x40065CFB: malloc (vg_clientmalloc.c:618)
=  =18081=  =  by 0x8048470: main (nasty.c:5)
=  =18081=  =  by 0x402D67EE: __libc_start_main (in /lib/libc.so.6)
```

```
=  =18081=  =  by 0x8048381: __libc_start_main@@GLIBC_2.0 (in /home/kalle/tmp/nasty)
=  =18081=  =
=  =18081=  =  LEAK SUMMARY:
=  =18081=  =     possibly lost: 0 bytes in 0 blocks.
=  =18081=  =     definitely lost: 0 bytes in 0 blocks.
=  =18081=  =     still reachable: 10 bytes in 1 blocks.
=  =18081=  =
```

Таким образом, Valgrind предоставляет не только очень полезный отладчик памяти, она также поставляется с несколькими так называемыми обертками. Одна из них – *cachegrind*, профилировщик, который в паре с графическим интерфейсом *kcachegrind* стал наиболее предпочтительным профилировщиком многих программистов. Во многих проектах *cachegrind* медленно вытесняет *gprof*.

Средства создания графического интерфейса

Ряд приложений и библиотек позволяют легко генерировать интерфейс пользователя для приложений, работающих в среде X Window System. Если вы не хотите утруждать себя сложностями программного интерфейса X, то альтернативой может быть использование одного из этих простых конструкторов интерфейса. Существуют и средства создания текстовых интерфейсов для программ, которые не требуют X.

В классической модели программирования X сделана попытка достичь максимальной общности, при этом вводится лишь самый минимум ограничений и допущений на интерфейс. Эта общность позволяет программистам создавать собственные интерфейсы с чистого листа, поскольку основные библиотеки X не делают предварительных допущений относительно интерфейса. X Toolkit Intrinsics (Xt) предоставляет рудиментарный набор графических элементов интерфейса (простые кнопки, полосы прокрутки и т. п.), а также общий интерфейс для создания собственных элементов графического интерфейса в случае необходимости. К сожалению, это может потребовать большого объема работы от программистов, которые предпочли бы использовать набор готовых интерфейсных процедур. Имеется ряд наборов элементов Xt и программных библиотек для Linux, облегчающих программирование интерфейса пользователя.

Qt, инструментальный набор C++ GUI, разработан норвежской фирмой Trolltech. Библиотека Qt является коммерческим продуктом, но он также выпускается под лицензией GPL, что означает возможность бесплатного использования при создании с ее помощью бесплатного программного обеспечения для UNIX (и, следовательно, Linux), которое тоже выпускается под лицензией GPL. Кроме того, есть коммерческие версии Qt для Windows и Mac OS X, что позволяет вести разработку одновременно для Linux, Windows и Mac OS X и создавать приложение для других платформ путем простой перекомпиляции. В результате предоставляется возможность вести разработку в своей любимой операционной системе Linux и все же быть в состоянии нацелиться на более обширный рынок Windows! Один из авторов, Калле, использует Qt для написания как бесплатного программного обеспечения (для упоминавшейся среды KDE), так и коммерческого (кроссплатформенные продукты, разрабатываемые для Linux, Windows и Mac OS X). Qt разрабатывается очень активно; более подробные сведения вы найдете в книге в «Programming with Qt» (O'Reilly).

Другим недавним замечательным дополнением к Qt является возможность ее работы на встроенных системах, не требующих X-сервера. А какую операционную систему будет она поддерживать на встроенных системах, если не Embedded Linux! Можно ожидать появления в скором будущем многих портативных устройств с графическими экранами, где будет работать Embedded Linux с Qt/Embedded, например в сотовых телефонах, автомобильных компьютерах, медицинской аппаратуре и во многих других устройствах. Нельзя сказать, что надпись «Linux» будет написана крупными буквами на коробке, но эта система будет внутри!

Вместе с Qt поставляется конструктор графического интерфейса пользователя под названием Qt Designer, значительно облегчающий создание приложений GUI. Он включен и в GPL-версию Qt, поэтому, загрузив Qt (или просто установив со своего дистрибутива), вы получите и Qt Designer. Наконец, существуют библиотеки, обеспечивающие возможность работы с библиотекой Qt в программах, написанных на языке Python, что позволяет создавать гибкие и привлекательные интерфейсы без необходимости изучать язык низкого уровня.

Для тех, кто не любит программировать на C++, хорошим выбором может стать библиотека GTK. Изначально эта библиотека разрабатывалась для программы графического редактора GIMP. Программы GTK обычно демонстрируют такое же хорошее время реакции, что и Qt, но сам набор инструментов не столь полон. Особенно ощущается недостаток документации. Однако проектам, основанным на языке C, GTK представляет хорошую альтернативу, если вы не собираетесь перекомпилировать свой код для Windows. Кроме того, была разработана версия GTK для Windows. Были созданы прекрасные библиотеки для работы с GTK из программ, написанных на языке C++, поэтому все большее число разработчиков GTK выбирают для своих разработок не язык C, а C++.

Многие программисты считают, что создание интерфейса пользователя, даже при наличии полного набора элементов и функций на языке C, является сложным и весьма трудоемким делом. Это вопрос соотношения гибкости и простоты программирования: чем проще создать интерфейс, тем меньшим контролем над ним обладает программист. Многие программисты считают, что готовые элементы удовлетворяют их потребностям, поэтому потеря гибкости не является большой проблемой.

В 1980-е и 1990-е годы наибольшей популярностью пользовалась коммерческая библиотека и набор элементов графического интерфейса – Motif, с недорогой лицензией для отдельных пользователей. Существует масса приложений, использующих Motif. Двоичные файлы, статически связанные с Motif, могут свободно распространяться и использоваться теми, кто не имеет Motif в своей собственности. В последние годы библиотека Motif находит применение разве что в проектах с давней историей – большинство программистов перешли на использование современных инструментальных средств Qt и GTK. Однако Motif по-прежнему активно разрабатывается (хотя и довольно медленно), но с этой библиотекой связаны свои проблемы. Во-первых, программирование на Motif не очень привлекательно. Связано это с тем, что оно отличается сложностью, возможностью совершения ошибок и утомительно, так как прикладной интерфейс Motif создавался без учета современных принципов проектирования графических интерфейсов. Во-вторых, созданные с помощью Motif программы выполняются очень медленно.

Одной из проблем при создании интерфейса и программировании для X является то, что трудно обобщить наиболее часто используемые элементы интерфейса пользователя в простой программной модели. Например, многие программы используют такие объекты, как кнопки, диалоговые окна, раскрывающиеся меню и т. д., но почти все программы используют эти элементы в разном контексте. При упрощении задачи создания графического интерфейса средства генерации стараются предположить, что вам потребуется. Например, достаточно указать, что при нажатии кнопки должна выполняться некоторая процедура в программе. Но как быть, если вы хотите, чтобы при нажатии кнопки осуществлялось некоторое особое поведение, не предусмотренное интерфейсом программирования? Например, если вы хотите, чтобы нажатие левой и правой кнопок мыши приводило к разным результатам? Если система создания интерфейса не допускает такой степени общности, то от нее мало пользы программистам, которым требуется мощный настраиваемый интерфейс.

Комплект Tcl/Tk, состоящий из языка сценариев Tcl и инструментального набора графики Tk, приобрел некоторую популярность отчасти потому, что он прост в использовании и обеспечивает большую гибкость. Поскольку функции Tcl и Tk можно вызывать из интерпретируемых «сценариев», а также из программ на языке C, нетрудно связать функции интерфейса, предоставляемые этим языком и инструментальным набором, с функционированием программы. Использование Tcl и Tk в целом требует меньших затрат, чем изучение непосредственного программирования с использованием Xlib и Xt (а также множества наборов элементов). Однако следует заметить, что чем больше проект, тем более вероятно, что вам потребуется использовать такой язык, как C++, более пригодный для крупномасштабных разработок. По ряду причин крупные проекты становятся очень громоздкими, когда привлекается Tcl: использование интерпретируемого языка снижает скорость исполнения программы, архитектуру Tcl/Tk трудно масштабировать на большие проекты, отсутствуют важные средства обеспечения надежности, такие как проверка типов на этапах компиляции и компоновки. Проблему масштабируемости можно решить благодаря использованию пространств имен (способ предотвращения конфликтов имен в различных частях программы) и объектно-ориентированного расширения под названием [incr Tcl].

Tcl и Tk позволяют создавать интерфейс для X, укомплектованный окнами, кнопками, меню, полосами прокрутки и прочим, вокруг уже существующих программ. Доступ к интерфейсу может осуществляться из сценария Tcl (как описано в разделе «Другие языки» далее в этой главе) или из программы на языке C.

Если вам требуется красивый текстовый интерфейс для программы, то имеется несколько возможностей на выбор. Библиотека GNU *getline* является набором функций, обеспечивающим развитое редактирование командной строки, вывод приглашений, буфер команд и другие функции, используемые во многих программах. Например, как *bash*, так и *gdb* используют библиотеку *getline* для ввода команд пользователя. *getline* предоставляет функции редактирования командной строки в стиле Emacs и *vi*, которые есть в *bash* и аналогичных программах. (Использование редактирования командной строки в *bash* описано в разделе «Сокращенный ввод с клавиатуры» главы 4.)

Еще одна возможность – написать ряд функций для взаимодействия вашей программы с Emacs. Примером является интерфейс *gdb* Emacs, в котором создается

несколько окон, устанавливаются специальные последовательности клавиш и т. д. Этот интерфейс обсуждался выше в разделе «Использование Emacs с gdb». (Для его реализации не потребовалось изменений в коде *gdb*: посмотрите в библиотеке Emacs в файле *gdb.el* заметки по поводу того, как это было сделано.) Emacs позволяет запустить подпрограмму из текстового буфера и предоставляет много подпрограмм для синтаксического анализа и обработки текста в этом буфере. Например, в интерфейсе Emacs для *gdb* выдаваемый отладчиком *gdb* листинг исходного текста перехватывается редактором Emacs и превращается в команду, отображающую текущую строку кода в другом окне. Процедуры, написанные на Emacs LISP, обрабатывают выходные данные, получаемые от *gdb*, и осуществляют некоторые действия, основываясь на ней.

Преимущество использования Emacs для взаимодействия с текстовыми программами состоит в том, что Emacs сам по себе является мощным и настраиваемым интерфейсом пользователя. Пользователь может легко переопределять клавиши и команды, приспосабливая их к своим потребностям, – вам не нужно самостоятельно обеспечивать эти функции настройки. Если текстовый интерфейс программы достаточно прост, чтобы взаимодействовать с Emacs, настройку выполнить несложно. Кроме того, многие пользователи предпочитают делать в Emacs буквально все – от чтения электронной почты и телеконференций до компиляции и отладки программ. Если вы снабдите свою программу интерфейсом Emacs, это облегчит ее использование людьми подобного склада ума. Это также позволит вашей программе взаимодействовать с другими программами, работающими под Emacs. Например, можно легко вырезать и переносить текст между различными текстовыми буферами Emacs. При желании можно писать целые программы на Emacs LISP.

Средства контроля ревизий – RCS

На Linux перенесена Revision Control System (RCS) – система контроля ревизий, представляющая собой комплект программ, позволяющих поддерживать «библиотеку» файлов, записывать историю ревизий, осуществлять блокировку файлов с исходным кодом (в случае работы нескольких людей над одним и тем же проектом) и автоматически отслеживать номера версий файлов с исходными текстами. RCS обычно используется с файлами исходного кода программы, но благодаря своей общности может применяться к любому типу файлов, для которых должны поддерживаться различные версии.

Зачем нужен контроль версий? Во многих больших проектах требуется тот или иной контроль ревизий для отслеживания многих мелких изменений в системе. Например, попытка разработки программы, состоящей из тысячи исходных файлов, бригадой из нескольких десятков программистов практически невозможна без использования какого-либо средства, аналогичного RCS. RCS в каждый момент разрешает модифицировать любой исходный файл только одному человеку, а все изменения вносятся в систему вместе с их описанием в журнале.

RCS основывается на понятии *файла RCS*, выполняющего функции «библиотеки», в которую файлы «сдаются» (check in) и из которой они «выдаются» (check out). Допустим, у вас есть исходный файл *importrtf.c*, который вы хотите поддерживать с помощью RCS. По умолчанию файл получает в RCS имя *importrtf.c,v*. Файл RCS содержит историю ревизий файла, что позволяет извлекать любую

предшествующую версию файла, внесенную в него. Для каждой версии вы можете сделать запись в журнале.

При сдаче файлов в RCS ревизии добавляются к файлу RCS, а исходный файл по умолчанию удаляется. Чтобы получить доступ к первоначальному файлу, нужно выписать его из файла RCS. Если вы редактируете файл, то вряд ли хотите, чтобы одновременно с вами кто-либо еще редактировал этот же файл. Поэтому RCS блокирует файл на то время, когда он выдан для редактирования. Заблокированный файл может быть модифицирован только тем пользователем, который его выписал (это реализуется через права доступа к файлу). Сделав изменения в исходном файле, вы сдаете его обратно, что позволяет всем работающим над проектом, в свою очередь, получить его для дальнейшей работы. Получение файла без его блокировки не влечет таких ограничений. Обычно файлы выписываются с блокировкой только для редактирования, а просто для чтения – без блокировки (например, для использования исходного файла при компиляции программы).

RCS автоматически отслеживает все предыдущие версии в файле RCS и присваивает возрастающие номера версий каждой новой сдаваемой ревизии. При сдаче файла можно присвоить собственный номер версии, что позволяет начать новую «ветвь» ревизий. При этом несколько проектов могут развиваться, исходя из разных версий одного и того же файла. Это удобный способ использовать в проектах общий код и при этом быть уверенным, что изменения в одной ветви не затрагивают другие.

Приведем пример. Возьмем исходный файл *importrtf.c*, содержащий нашу дружелюбную программу:

```
#include <stdio.h>

int main(void) {
    printf("Hello, world!");
}
```

Сначала нужно сдать его в систему RCS с помощью команды *ci*:

```
papaya$ ci importrtf.c
importrtf.c,v <-- importrtf.c
enter description, terminated with single '.' or end of file:
NOTE: This is NOT the log message!
>> Hello world source code
>> .
initial revision: 1.1
done
papaya$
```

Создан файл RCS *importrtf.c,v* и удален файл *importrtf.c*.

Чтобы снова начать работу с файлом, содержащим исходный текст программы, нужно выполнить команду *co* («check out»). Например, следующая команда

```
papaya$ co -l importrtf.c
importrtf.c,v --> importrtf.c
revision 1.1 (locked)
done
papaya$
```

выдает файл *importrtf.c* (из *importrtf.c,v*) и блокирует его. Блокировка файла позволяет отредактировать его и сдать обратно. Если файл нужен вам только для чтения (например, чтобы запустить *make*), можно опустить ключ *-l* в команде *co*, и файл останется незаблокированным. Нельзя сдать файл, который не заблокирован (если только это не новый файл, как в нашем примере).

Теперь можно сделать изменения в исходном коде и сдать файл обратно. В большинстве случаев желательно постоянно иметь файл «на руках», а команду *ci* использовать для записи последней ревизии в файл RCS и выдачи номера версии. Для этого можно использовать в команде *ci* ключ *-l*:

```
papaya$ ci -l importrtf.c
importrtf.c,v <-- importrtf.c
new revision: 1.2; previous revision: 1.1
enter log message, terminated with single '.' or end of file:
>> Changed printf call
>> .
done
papaya$
```

В результате после сдачи файл автоматически выдается с блокировкой. Это полезный способ контроля ревизий, даже если над проектом работает один человек.

Если вы часто используете RCS, то нагромождение в вашем каталоге всех этих файлов *importrtf.c,v* может выглядеть неопрятно. Создайте подкаталог RCS в вашем каталоге проекта, тогда команды *ci* и *co* будут помещать файлы RCS в него, не перемешивая их с файлами, содержащими остальные исходные тексты.

Кроме того, RCS отслеживает все прежние ревизии вашего файла. Например, если произведенные в программе изменения приводят ее к аварии и вы хотите отменить ваши изменения, вернувшись к прежней версии, можно указать конкретный номер версии, которую должна выдать команда *co*. Например, команда

```
papaya$ co -l1.1 importrtf.c
importrtf.c,v --> importrtf.c
revision 1.1 (locked)
writable importrtf.c exists; remove it? [ny](n): y
done
papaya$
```

выдаст версию 1.1 файла *importrtf.c*. Чтобы просмотреть историю ревизий некоторого файла, можно воспользоваться программой *rlog*, которая выводит записи журнала ревизий (введенные в команде *ci*), а также другие данные, такие как дата, имя пользователя, сдавшего ревизию.

При выдаче файла с исходным кодом RCS автоматически обновляет встроенные в него строки ключевых слов. Например, если в исходном файле есть строка:

```
/* $Header$ */
```

тогда *co* заменит ее информативной строкой с датой ревизии, номером версии и другой информацией, как показано ниже:

```
/* $Header: /work/linux/hitch/programming/tools/RCS/rcs.tex
       1.2 1994/12/04 15:19:31 mdw Exp mdw $ */
```

(На самом деле это одна строка; она была разбита на две, чтобы уместиться на книжной странице.)

Есть и другие ключевые слова, такие как `$Author$`, `$Date$` и `$Log$`.

Многие программисты помещают в исходные тексты статические строки для определения версии программы после ее компиляции. Например, в каждый файл с исходным текстом вашей программы можно поместить строку:

```
static char rcsid[] = "\@(#)$Header$";
```

тогда *co* заменит ключевое слово `$Header$` строкой приведенного вида. Эта статическая строка сохраняется в исполняемом файле и может быть выведена с помощью команды *what*. Например, скомпилировав *importrtf.c* в исполняемый файл *importrtf*, можно выполнить команду:

```
papaya$ what importrtf
importrtf:
        $Header: /work/linux/hitch/programming/tools/RCS/rcs.tex
                        1.2 1994/12/04 15:19:31 mdw Exp mdw $
papaya$
```

what находит в файле строки, начинающиеся с символов `@(#)`, и выводит их. Если у вас есть программа, скомпилированная из большого числа исходных файлов и библиотек, и вы не знаете, насколько новы составляющие ее компоненты, тогда с помощью *what* можно вывести строки версий для всех исходных файлов, использовавшихся при компиляции в двоичный файл.

В состав пакета RCS входят еще несколько программ, в том числе *rcs*, используемая для сопровождения файлов RCS. Помимо всего прочего, *rcs* может предоставлять другим пользователям право получения исходных файлов из файла RCS. За дополнительной информацией обращайтесь к страницам справочного руководства команд *ci*(1), *co*(1) и *rcs*(1).

Средства контроля ревизий – CVS

Система CVS (Concurrent Version System) более сложна, чем RCS, и поэтому, вероятно, чрезмерна для проектов, разрабатываемых одним человеком. Но если над проектом работают более одного-двух программистов или исходные тексты размещены в нескольких каталогах, CVS предпочтительнее. Для хранения изменений CVS использует формат файла RCS, но применяет собственную структуру управления.

По умолчанию CVS работает с целыми деревьями каталогов, то есть каждая команда CVS воздействует на текущий каталог и на все вложенные в него подкаталоги. Этот рекурсивный обход можно отключить с помощью параметра командной строки либо можно указать отдельный файл, с которым должна работать команда.

CVS формализовала понятие «песочницы», используемое многими компаниями, разрабатывающими программное обеспечение. Эта концепция предполагает наличие так называемого хранилища, или *репозитория* (*repository*), содержащего «официальные» исходные коды, относительно которых известно, что они компилируются и работают (по крайней мере, частично). Ни одному разработчику не позволяется непосредственно редактировать файлы в этом хранилище.

Вместо этого он выписывает файлы в локальное дерево каталогов, или *песочницу* (*sandbox*), где может редактировать исходные тексты, сколько душе угодно – вносить изменения, добавлять и удалять файлы и заниматься всем тем, чем обычно занимаются разработчики (нет, не играть в Quake или есть пончики). Удостоверившись, что сделанные им изменения компилируются и работают, разработчик передает файлы обратно в хранилище и делает их доступными другим разработчикам.

Когда вы получаете файлы в локальное дерево каталогов, они все имеют разрешение для записи. Все необходимые изменения вы производите в файлах личного рабочего пространства. Закончив тестирование и чувствуя уверенность в своей работе, достаточную для того, чтобы поделиться своей работой с остальными разработчиками, вы записываете измененные файлы в хранилище с помощью команды CVS *commit*. CVS проверяет, не сделаны ли каким-либо другим разработчиком изменения с тех пор, как вы получили свое дерево каталогов. В этом случае CVS не разрешает вам внести свои изменения, а предлагает принять изменения, относящиеся к вашему локальному дереву и сделанные другими разработчиками. Во время этой операции обновления CVS применяет сложный алгоритм в попытке согласовать («объединить») ваши изменения с изменениями, сделанными другими разработчиками. Не всегда это удается сделать автоматически. В этом случае CVS сообщает о наличии конфликтов и просит разрешить их. Спорный файл размечается специальными символами, для того чтобы вы могли видеть место конфликта и решить, какую из версий следует использовать. При этом CVS допускает наличие конфликтов только в локальных каталогах разработчиков, а в хранилище всегда лежит непротиворечивая версия.

Создание хранилища CVS

Если вы участвуете в большом проекте, то, вероятно, кто-то уже установил все необходимое для работы CVS. Но если вы сами администрируете проект и хотите повозиться с CVS на своей локальной машине, вам придется самостоятельно создать хранилище.

Сначала укажите в переменной окружения `CVSROOT` каталог, в котором вы хотите разместить хранилище CVS. CVS может поддерживать в хранилище любое количество проектов и обеспечивать их независимое существование, поэтому нужно лишь однажды выбрать каталог для хранения всех проектов, поддерживаемых CVS, и вам не придется изменять его при переходе от одного проекта к другому. Вместо переменной `CVSROOT` можно во всех командах CVS использовать ключ *-d*, но поскольку вводить каждый раз имя каталога утомительно, мы предполагаем, что вы установили переменную `CVSROOT`.

Если определен каталог для хранилища, можно создать само хранилище следующей командой (в предположении, что CVS установлена на вашей машине):

```
$tigger cvs init
```

Существует несколько способов создания дерева проекта в хранилище CVS. Если у вас уже есть дерево каталогов, но оно еще не управляется RCS, можно просто импортировать его в хранилище командой:

```
$tigger cvs import directory manufacturer tag
```

где *directory* является именем каталога самого верхнего уровня проекта, *manufacturer* – именем автора кода (вы можете использовать здесь любое имя), а *tag* является так называемым тегом релиза, который может быть выбран произвольно. Например:

```
$tigger cvs import dataimport acmeinc initial
... здесь программа выведет большой объем информации ...
```

Если вы хотите начать новый проект, можно просто создать дерево каталогов командами *mkdir*, а затем импортировать это пустое дерево, как показано в предыдущем примере.

Если вы хотите импортировать проект, уже управляемый RCS, возникают некоторые дополнительные сложности, поскольку использовать *cvs import* нельзя. В этом случае придется создать необходимые каталоги прямо в хранилище, а затем скопировать в эти каталоги все файлы RCS (все файлы, оканчивающиеся на *,v*). Не используйте здесь подкаталоги RCS!

В каждом хранилище есть файл с именем *CVSROOT/modules*, содержащий названия проектов хранилища. Для добавления нового модуля можно редактировать файл *modules*. Этот файл можно получать, редактировать и сдавать, как любой другой файл в хранилище. Таким образом, чтобы добавить к списку ваш модуль, выполните следующие команды:

```
$tigger cvs checkout CVSROOT/modules
$tigger cd CVSROOT
$tigger emacs modules
... или любой другой редактор; что вводить, сказано ниже ...
$tigger cvs commit modules
$tigger cd ..
$tigger cvs release -d CVSROOT
```

Если вы не делаете чего-либо необычного, то формат файла *modules* очень прост: каждая строка начинается с имени модуля, за которым следуют пробел или знак табуляции и путь внутри хранилища. В файле *modules* можно делать много других вещей, о которых рассказано в документации по CVS, доступной на сайте *http://www.loria.fr/~molli/cvs-index.html*. Есть также короткая, но очень содержательная книга «CVS Pocket Reference» Грегора Пурди (Gregor N. Purdy), изданная O'Reilly.

Работа с CVS

В этом разделе мы предполагаем, что вы или ваш системный администратор установили модуль с именем `dataimport`. Вы можете получить локальное дерево каталогов этого модуля командой:

```
$tigger cvs checkout dataimport
```

Если для проекта, над которым вы собираетесь работать, не определен модуль, вам нужно узнать путь внутри хранилища. Например, может потребоваться что-то вроде:

```
$tigger cvs checkout clients/acmeinc/dataimport
```

Какую бы версию команды *checkout* вы ни использовали, CVS создаст каталог с именем *dataimport* в вашем текущем рабочем каталоге и выпишет в него из

хранилища все файлы и каталоги, относящиеся к этому модулю. Все файлы доступны для записи, и вы можете немедленно начать их редактирование.

Произведя какие-то изменения, вы можете записать измененные файлы обратно в хранилище, выполнив единственную команду:

```
$tigger cvs commit
```

Конечно, можно записывать в хранилище и отдельные файлы:

```
$tigger cvs commit importrtf.c
```

Что бы вы ни делали, CVS так же, как это делает RCS, попросит ввести комментарии, которые должны быть введены вместе с вашими изменениями. Но CVS предоставляет дополнительные удобства по сравнению с RCS. Вместо простого приглашения RCS вы получите полноэкранный редактор для ввода. Используемый редактор можно выбрать, установив переменную окружения `CVSEDITOR`. Если она не установлена, CVS ищет редактор в `EDITOR`, а если он не определен и там, CVS вызывает *vi*. Если вы выписали проект целиком, CVS будет использовать введенный вами комментарий для каждого каталога, в котором были изменения, но каждый раз будет заново запускать редактор, чтобы вы могли сделать изменения в комментарии.

Как уже говорилось, нет необходимости корректно устанавливать `CVSROOT` для получения файлов, поскольку при записи дерева в хранилище CVS создала каталог CVS в каждом рабочем каталоге. Этот каталог содержит все данные, необходимые CVS для работы, в том числе путь, по которому находится хранилище.

Вполне может оказаться, что пока вы работали над своими файлами, ваш коллега записал в хранилище некоторые из файлов, над которыми вы в настоящее время трудитесь. В этом случае CVS не позволит вам сохранить ваши файлы, а предложит сначала обновить ваше локальное дерево. Сделайте это командой:

```
$tigger cvs update
M importrtf.c
A exportrtf.c
? importrtf
U importword.c
```

(Можно также указать конкретный файл.) Внимательно изучите вывод этой команды: CVS выводит имена всех обрабатываемых файлов, каждому из которых предшествует один символ. Этот символ сообщает о том, что произошло во время операции обновления. Наиболее важные символы приведены в табл. 21.1.

Таблица 21.1. Ключевые символы для файлов в CVS

Символ	Описание
`P`	Файл был обновлен. Символ `P` выводится, если файл уже был добавлен в хранилище или был изменен, но сами вы никаких изменений в этот файл не вносили.
`U`	Вы изменили этот файл, но никто другой этого не делал.
`M`	Вы изменили этот файл и кто-то другой тоже записал более новую версию. Все изменения удалось благополучно объединить.

Символ	Описание
C	Вы изменили этот файл и кто-то другой тоже записал более новую версию. При попытке объединения изменений возникли конфликты.
?	В CVS нет сведений об этом файле, то есть он не контролируется CVS.

Наиболее важным из этих символов является C: CVS не смогла объединить изменения и нуждается в вашей помощи. Загрузите файлы в редактор и поищите строку <<<<<<<. За этой строкой снова показывается имя файла, за которым следует ваша версия, заканчивающаяся строкой, содержащей =======. Затем следует версия кода из хранилища, которая заканчивается строкой, содержащей >>>>>>>. Теперь вы должны установить, возможно, связавшись со своим коллегой, какая версия лучше и можно ли объединить две версии вручную. Измените файл должным образом и удалите метки CVS <<<<<<<, ======= и >>>>>>>. Сохраните файл и еще раз выполните *commit*.

Если вы решили временно прекратить работу над проектом, проверьте, все ли изменения внесены в хранилище. Для этого перейдите в каталог над корневым каталогом вашего проекта и выполните команду:

```
$tigger cvs release dataimport
```

CVS проверит, внесли ли вы все изменения обратно в хранилище, и при необходимости предупредит вас. Полезно использовать параметр *-d*, который удаляет локальное дерево, если все изменения внесены в хранилище.

Работа с CVS через Интернет

CVS очень удобна для рассредоточенных групп разработчиков, поскольку предоставляет возможность доступа к хранилищам, находящимся на других машинах.[1]

В настоящее время есть как бесплатные (например, SourceForge), так и коммерческие сервисы, предоставляющие сервер CVS, благодаря чему вы можете начать сетевой программный проект, не заводя собственного круглосуточно работающего сервера.

Если вы можете с помощью *rsh* подключиться к машине, на которой находится хранилище, то у вас есть возможность использовать для доступа к хранилищу удаленную систему CVS. Чтобы выписать из хранилища модуль, выполните команду:

```
cvs -d :ext:user@domain.com:/path/to/repository checkout dataimport
```

Если вы не можете или не хотите использовать *rsh* по соображениям безопасности, можно использовать защищенную оболочку *ssh*. Вы можете сообщить CVS, что хотите использовать *ssh*, установив значение переменной окружения CVS_RSH в *ssh*.

Аутентификацию и доступ к хранилищу можно также осуществить по протоколу «клиент-сервер». Для удаленного доступа необходимо, чтобы на машине с храни-

[1] Использование CVS расцвело с ростом числа проектов свободного программного обеспечения, разрабатываемого с помощью Интернета людьми с разных континентов.

лищем работал сервер CVS. О том, как это сделать, читайте в документации по CVS. Если сервер установлен, подключиться к нему можно следующим образом:

```
cvs -d :pserver:user@domain.com:path/to/repository login
CVS password:
```

Как видно из этого примера, сервер CVS запросит пароль CVS, назначенный вам администратором сервера CVS. Эта процедура регистрации требуется только один раз для каждого хранилища. При получении модуля нужно указать машину, на которой находится сервер, ваше имя пользователя на этой машине и удаленный путь к хранилищу. Как и для локальных хранилищ, эти данные сохраняются в вашем локальном дереве. Поскольку пароль сохраняется в файле *.cvspass* в вашем исходном каталоге с минимальной криптографической защитой, возникает потенциальная угроза безопасности. В документации CVS об этом сказано более подробно.

При использовании CVS в Интернете и загрузке больших модулей может понадобиться использование параметра *-z* с дополнительным числовым значением, определяющим степень сжатия передаваемых данных в диапазоне от 1 до 9.

Как уже отмечалось во вводном разделе главы, система CVS потихоньку вытесняется другой системой контроля версий – *Subversion*, однако CVS до сих пор используется большинством проектов, именно по этой причине мы и рассказываем о CVS так подробно. Тем не менее один из крупнейших проектов с открытыми исходными текстами, KDE, перешел на использование *Subversion*, и теперь, как ожидается, его примеру последуют многие другие проекты. Наборы команд этих двух систем очень похожи, например, при использовании *Subversion* добавление файла в репозиторий производится командой *svn add* вместо *cvs add*. Главное преимущество *Subversion* перед CVS состоит в том, что запись изменений командой *commit* производится атомарно, то есть либо будут отправлены все измененные файлы, либо ни одного (тогда как CVS гарантирует подобное поведение только для одного каталога). Вследствие этого система *Subversion* хранит не номера версий для отдельных файлов, как CVS, а номера версий модулей, что упрощает возможность извлечения целого пакета исходных текстов. Дополнительные сведения о *Subversion* можно найти на сайте *http://subversion.tigris.org*.

Обновление файлов с помощью patch

Допустим, вы пытаетесь сопровождать программу, которая периодически обновляется, но содержит столько исходных файлов, что невозможно при каждом обновлении выпускать полный дистрибутив исходных текстов. Лучше всего производить инкрементное обновление исходных файлов с помощью программы *patch*, созданной Ларри Уоллом (Larry Wall), автором языка программирования Perl.

Программа *patch* производит в файле контекстно-зависимые изменения с целью обновления его до следующей версии. Поэтому при изменении программы вы просто выпускаете файл-заплатку («патч») для исходного файла, который пользователь применяет с помощью *patch* и получает свежую версию. Например, Линус Торвальдс обычно выпускает новые версии ядра и в виде заплаток, и как полные дистрибутивы.

Замечательной особенностью *patch* является то, что она применяет обновления в контексте. Это означает, что если вы самостоятельно изменили исходный код, но

все же хотите использовать изменения, предлагаемые файлом-заплаткой, *patch* обычно в состоянии правильно определить то место в исходном файле, которое нужно изменить. Благодаря этому ваши версии оригинальных файлов с исходным кодом не должны точно совпадать с теми, для которых создавалась заплатка.

Для создания файла-заплатки используется программа *diff*, которая вычисляет «контекстные различия» между двумя файлами. Возьмем, например, исходный код нашей программы «Hello World!»:

```
/* hello.c version 1.0 by Norbert Ebersol */
#include <stdio.h>

int main() {
    printf("Hello, World!");
    exit(0);
}
```

Допустим, мы хотим обновить этот код, чтобы он стал таким:

```
/* hello.c version 2.0 */
/* (c)1994 Norbert Ebersol */
#include <stdio.h>

int main() {
    printf("Hello, Mother Earth!\n");
    return 0;
}
```

Если вы хотите создать файл-заплатку для обновления исходного текста *hello.c* до новейшей версии, выполните *diff* с параметром *-c*:

```
papaya$ diff -c hello.c.old hello.c > hello.patch
```

В результате будет создан файл-заплатка *hello.patch*, в котором описано, как преобразовать исходный текст *hello.c* (сохраненный в нашем случае в файле *hello.c.old*) до новой версии. Вы можете поставлять этот файл-заплатку всем, у кого есть оригинальная версия «Hello, World!», для проведения обновления с помощью *patch*.

Использование *patch* отличается простотой: в большинстве случаев вы просто запускаете ее с файлом-заплаткой на входе:[1]

```
papaya$ patch < hello.patch
Hmm... Looks like a new-style context diff to me...
The text leading up to this was:
--------------------------
|*** hello.c.old        Sun Feb 6 15:30:52 1994
|--- hello.c     Sun Feb 6 15:32:21 1994
--------------------------
Patching file hello.c using Plan A...
Hunk #1 succeeded at 1.
```

[1] Результаты работы, которые здесь приводятся, были получены с помощью последней версии утилиты *patch*, выпущенной Ларри Уоллом (Larry Wall), – версии 2.1. Если у вас более новая версия *patch*, то для получения аналогичных результатов нужно использовать флаг *--verbose*.

```
done
papaya$
```

patch выдает предупреждение, если ей кажется, что заплатка уже применялась. Если попытаться применить заплатку еще раз, *patch* спрашивает, не предполагалось ли задать ключ *-R*, который отменяет установленную заплатку. Это хороший способ отмены заплаток, которые вы не собирались применять. *patch* также сохраняет исходные версии обновляемых файлов в виде резервных копий, обычно именуемых как *filename~* (имя файла с приписанной тильдой).

Во многих случаях требуется обновить не отдельный файл, а файлы из целого дерева каталогов. *patch* позволяет обновлять группы файлов исходя из одного файла-заплатки, полученного *diff*. Допустим, что у вас есть два дерева каталогов – *hello.old* и *hello*, в которых содержатся исходные файлы для старой и новой версий программы, соответственно. Чтобы сделать файл заплатки для целого дерева, используйте ключ *-r*:

```
papaya$ diff -cr hello.old hello > hello.patch
```

Теперь посмотрим, что делается в системе, программное обеспечение в которой нуждается в обновлении. Предполагая, что исходный текст содержится в каталоге *hello*, можно применить заплатку такой командой:

```
papaya$ patch -p0 < hello.patch
```

Ключ *-p0* сообщает *patch* о необходимости сохранить пути обновляемых файлов (поэтому программа ищет исходные файлы в каталоге *hello*). Если исходные тексты, которые нужно обновить, находятся не в том каталоге, который указан в файле-заплатке, можно воспользоваться параметром *-p* без числа. Подробности ищите на страницах справочного руководства по *patch*(1).

Форматирование исходных текстов

Если вы озабочены расстановкой отступов в коде, а редакторы, автоматически устанавливающие отступы во вводимом тексте «на лету», несколько раздражают вас, можно воспользоваться программой *indent* для красивого оформления исходных текстов по окончании его ввода. *indent* является интеллектуальным форматером программ, написанных на языке С, и обладает большим количеством параметров, позволяющих указать желательный стиль отступов.

Рассмотрим пример ужасно отформатированного исходного кода:

```
double fact (double n) { if (n==1) return 1;
else return (n*fact(n-1)); }
int main () {
printf("Factorial 5 is %f.\n",fact(5));
printf("Factorial 10 is %f.\n",fact(10)); exit (0); }
```

В результате применения *indent* к этому исходному коду получим довольно красивый код:

```
#include <math.h>

double
fact (double n)
```

```
{
  if (n == 1)
    return 1;
  else
    return (n * fact (n - 1));
}
void
main ()
{
  printf ("Factorial 5 is %f.\n", fact (5));
  printf ("Factorial 10 is %f.\n", fact (10));
  exit (0);
}
```

Здесь не только хорошо выставлены отступы строк, но и добавлены пробелы вокруг операторов и параметров функций, что делает их удобочитаемыми. Существует много способов определить, как должны выглядеть результаты работы *indent*. Если данный стиль отступов вам не нравится, *indent* может его изменить.

indent может также создавать из исходного файла код для *troff*, который пригоден для печати или включения в технический документ. Он будет обладать такими приятными характеристиками, как выделение комментариев курсивом, выделение ключевых слов полужирным шрифтом и т. д. Команда

```
papaya$ indent -troff importrtf.c | groff -mindent
```

создаст код для *troff* и отформатирует его с помощью *groff*.

Наконец, *indent* можно использовать в качестве простого средства отладки. Если вы поместили } в неправильном месте, то, пропустив программу через *indent*, вы увидите, как компьютер представляет себе структуру блоков в вашей программе.

Использование Perl

Появление Perl, возможно, является самым благодатным событием в мире программирования для UNIX за многие годы. Его ценность не умаляется доступностью только в Linux.[1] Perl ориентирован на работу с текстами и файлами и первоначально был создан для просмотра и обработки больших объемов текста, а также для создания красивых отчетов из этих данных. Однако по мере своего становления Perl превратился в язык сценариев общего назначения, пригодный для решения любых задач от управления процессами до сетевых взаимодействий по TCP/IP. Perl, являясь свободным программным обеспечением, был разработан Ларри Уоллом (Larry Wall), гуру UNIX, которому мы обязаны существованием программы чтения телеконференций *rn* и многих распространенных утилит, таких как *patch*. В настоящее время его поддерживают Ларри и группа добровольцев.

Основная сила Perl заключается в том, что он объединил наиболее широко используемые возможности таких языков, как C, *sed*, *awk*, и различных оболочек

[1] Справедливости ради следует отметить, что теперь Perl есть и на других системах, в том числе Windows. Но его популярность и общеупотребительность там не идут ни в какое сравнение с тем, как им пользуются в Linux.

в один интерпретируемый язык сценариев. В прошлом, для того чтобы выполнить сложную работу, требовалось создавать из этих языков «художественные» конструкции, в которых зачастую сценарии на *sed* передавали данные в сценарии на *awk*, которые, в свою очередь, передавали результат в сценарии оболочки, а в конечном итоге – в программы на C. Perl освобождает от обычной философии UNIX, заключающейся в использовании многочисленных маленьких программ для обработки мелких частей большой задачи. На Perl делается вся задача, причем есть возможность выполнить ее несколькими способами. Фактически эта глава была написана программой искусственного интеллекта, разработанной на Perl. (Это шутка, Ларри.)

Perl предоставляет удобный программный интерфейс для многих функций, которые зачастую трудно использовать в других языках. Например, распространенной задачей, решаемой сценариями системного администрирования в UNIX, является просмотр большого объема текста, выбор полей из каждой строки текста на основе шаблона (обычно представленного в виде *регулярного выражения*) и составление отчета по этим данным. Допустим, вы хотите обработать вывод UNIX-команды *last*, которая отображает журнал времени регистрации всех пользователей системы в таком виде:

```
mdw      ttypf   loomer.vpizza.co Sun Jan 16 15:30 - 15:54 (00:23)
larry    ttyp1   muadib.oit.unc.e Sun Jan 16 15:11 - 15:12 (00:00)
johnsonm ttyp4   mallard.vpizza.c Sun Jan 16 14:34 - 14:37 (00:03)
jem      ttyq2   mallard.vpizza.c Sun Jan 16 13:55 - 13:59 (00:03)
linus    FTP     kruuna.helsinki. Sun Jan 16 13:51 - 13:51 (00:00)
linus    FTP     kruuna.helsinki. Sun Jan 16 13:47 - 13:47 (00:00)
```

Если нужно просуммировать общее время регистрации каждого пользователя в системе (приведенное в скобках в последнем поле), то можно написать сценарий на *sed*, который выберет значения времени, поданные на вход, затем написать сценарий на *awk*, который отсортирует данные по пользователям и просуммирует время, и еще один сценарий на *awk*, который создаст из полученных данных отчет. Можно также написать довольно сложную программу на C, которая выполнит всю работу. Программа будет сложной, поскольку всем программистам на C известно, что функции обработки текста в этом языке не сильно развиты.

Эта задача легко решается простым сценарием Perl. Средства ввода-вывода, поиск по регулярным выражениям, сортировка по ассоциативным массивам и числовые задачи – все это с легкостью достигается в программах на Perl. Программы на Perl, как правило, коротки и конкретны и не фетишизируют технические проблемы на пути к тому, что ваша программа действительно должна делать.

Использование Perl в Linux не отличается от использования в других системах UNIX. Имеется несколько хороших книг по Perl, изданных O'Reilly: «Programming Perl»[1] Ларри Уолла (Larry Wall), Тома Кристиансена (Tom Christiansen) и Джона Орванта (Jon Orwant), «Learning Perl»[2] Рэндала Л. Шварца (Randal L.

---

[1] Ларри Уолл, Том Кристиансен, Джон Орвант «Программирование на Perl», 3-е издание. – Пер. с англ. – СПб.: Символ-Плюс, 2002.

[2] Рэндал Шварц и Том Феникс «Изучаем Perl», 3-е издание. – Пер. с англ. – СПб: Питер, 2002.

Schwartz) и Тома Феникса (Tom Phoenix), «Advanced Perl Programming» Шрирама Шринивасана (Sriram Srinivasan) и «Perl Cookbook»[1] Тома Кристиансена и Натана Торкингтона (Nathan Torkington). Тем не менее мы считаем, что Perl служит таким прекрасным инструментом, что заслуживает некоторого введения. В конце концов, Perl является таким же свободно распространяемым программным обеспечением, как и Linux; они идут рука об руку.

Пример программы

Что нам действительно нравится в Perl, так это возможность сразу перейти к решаемой задаче. Нет необходимости писать пространный код для настройки структур данных, открытия файлов и каналов, отводить память под данные и т. д. Обо всем этом за вас позаботятся, причем самым дружественным образом.

Для знакомства с рядом основных функций Perl вернемся к примеру с учетом времени регистрации. Сначала мы представим сценарий целиком (снабженный комментариями), а затем опишем, как он работает. Этот сценарий читает выходные данные команды *last* (приводившиеся в предыдущем примере) и выводит запись по каждому пользователю системы, содержащую общее время работы в системе и число регистраций. (Номера строк слева проставлены для удобства описания):

```
1     #!/usr/bin/perl
2
3     while (<STDIN>) { # Пока есть входные данные...
4         # Найти строки и запомнить имя пользователя и время регистрации
5         if (/^(\S*)\s*.*\((.*):(.*)\)$/) {
6             # Нарастить сумму часов, минут и число регистраций
7             $hours{$1} += $2;
8             $minutes{$1} += $3;
9             $logins{$1}++;
10        }
11    }
12
13    # Для каждого пользователя в массиве ...
14    foreach $user (sort(keys %hours)) {
15        # Вычислить часы по сумме минут
16        $hours{$user} += int($minutes{$user} / 60);
17        $minutes{$user} %= 60;
18        # Вывести данные по пользователю
19        print "User $user, total login time ";
20        # В Perl тоже есть printf
21        printf "%02d:%02d, ", $hours{$user}, $minutes{$user};
22        print "total logins $logins{$user}.\en";
23    }
```

В строке 1 загрузчику сообщается, что сценарий должен выполняться через Perl, а не командным процессором. Программа начинается в строке 3. Это заголовок простого цикла `while`, с которым должны быть знакомы программисты, использующие язык C и командную оболочку: код в скобках в строках 4–10 дол-

[1] Том Кристиансен, Натан Торкингтон «Perl. Сборник рецептов. Для профессионалов», 2-е издание. – Пер. с англ. – СПб.: Питер, 2004.

жен выполняться, пока значение некоторого выражения истинно. Однако условное выражение <STDIN> выглядит любопытно. В действительности выражение читает одну строку со стандартного устройства ввода (представляемого в Perl именем STDIN) и делает эту строку доступной программе. Значение выражения истинно, пока есть входные данные.

Perl читает входные данные построчно (если не указано иное). По умолчанию ввод производится со стандартного устройства ввода, опять-таки если не задано иное. Поэтому в данном цикле while будут непрерывно читаться данные со стандартного устройства ввода, пока не останется строк для чтения.

То, что вы видите в строке 5, является всего лишь оператором if. Как и в большинстве языков программирования, код в скобках (строки 6–9) будет выполнен, если выражение, следующее за if, истинно. Но что представляет собой выражение внутри скобок? Читатели, знакомые с такими утилитами UNIX, как *grep* и *sed*, сразу определят его как регулярное выражение: загадочный, но эффективный способ представить шаблон, который должен быть найден во входном тексте. Регулярные выражения обычно ограничиваются с обеих сторон символами слэша (/. . . /).

Данное регулярное выражение соответствует строкам вида:

```
mdw     ttypf    loomer.vpizza.co Sun Jan 16 15:30 - 15:54 (00:23)
```

Это выражение также «запоминает» имя пользователя (*mdw*) и общее время регистрации в системе для этой записи (00:23). Само выражение вас пусть не беспокоит: составление регулярных выражений – сложная тема. Пока же вам достаточно знать, что данный оператор if находит строки по образцу, приведенному в примере, и вычленяет имя пользователя и время работы для дальнейшей обработки. Имя пользователя присваивается переменной $1, часы – переменной $2 и минуты – переменной $3. (В Perl переменные начинаются с символа $, но, в отличие от командной оболочки, символ $ должен также использоваться и при присваивании значения переменной.) Это присваивание осуществляет само регулярное выражение при нахождении соответствия. (Все, что заключено в регулярном выражении в скобки, сохраняется в переменных от $1 до $9 и может быть использовано в дальнейшем.)

В строках 6–9 происходит фактическая обработка этих данных. Она происходит интересным образом – через использование *ассоциативного массива*. В то время как обычный массив индексируется числовым индексом, ассоциативный массив индексируется произвольной строкой. Ассоциативные массивы находят применение во многих мощных приложениях, позволяя связывать один набор данных с другим, собираемым динамически. В нашей короткой программе ключами служат имена пользователей, получаемые из выходных данных *last*. Мы храним три ассоциативных массива, каждый из которых индексируется именем пользователя: hours, хранящий суммарное количество часов, в течение которых пользователь был зарегистрирован в системе; minutes, куда записывается количество минут, и logins, куда записывается общее число регистраций.

Например, ссылка на переменную $hours{'mdw'} возвращает общее количество часов, которое пользователь *mdw* провел в системе. Если имя пользователя *mdw* хранится в переменной $1, то ссылка на $hours{$1} имеет то же значение.

В строках 6–9 мы увеличиваем значения в этих массивах в соответствии с данными в текущей введенной строке. Например, если введена строка:

```
jem     ttyq2    mallard.vpizza.c Sun Jan 16 13:55 - 13:59 (00:03)
```

то строка 7 увеличит значение `hours`, индексированное `$1` (пользователь с именем *jem*), на количество часов, в течение которых пользователь *jem* был зарегистрирован (хранится в переменной `$2`). В языке Perl оператор инкрементирования `+=` эквивалентен соответствующему оператору языка C. В строке 8 аналогичным образом инкрементируется число минут для соответствующего пользователя. В строке 9 на единицу увеличивается значение счетчика регистраций с использованием оператора `++`.

Ассоциативные массивы являются одной из наиболее удобных возможностей языка Perl. Они позволяют создавать сложные базы данных во время синтаксического анализа текста. Было бы почти невозможно использовать с этой целью обычные массивы. Пришлось бы сначала подсчитать количество различных пользователей во входном потоке, а затем выделить для массива память соответствующего размера, назначив каждому пользователю место в массиве (используя хеширование или иную схему индексации). Ассоциативный же массив позволяет индексировать данные непосредственно с помощью строк, не обращая внимания на размер массива. (Конечно, при работе с большими массивами возникают проблемы скорости обработки, но для большинства приложений они несущественны.)

Пойдем дальше. В строке 14 использован оператор Perl `foreach`, который вам может быть знаком, если вы пишете сценарии командной оболочки. (Цикл `foreach` на самом деле сводится к циклу `for`, сходному с имеющимся в языке C.) При каждом проходе цикла переменной `$user` присваивается очередное значение из списка, заданного выражением `sort(keys %hours)`. `%hours` ссылается на весь ассоциативный массив `hours`, который мы создали. Функция `keys` возвращает список всех ключей, использованных для индексирования массива, который в данном случае является списком имен пользователей. Наконец, функция `sort` сортирует список, возвращенный `keys`. В результате цикл перебирает отсортированный список имен пользователей, поочередно присваивая имя пользователя переменной `$user`.

Строки 16 и 17 просто корректируют положение, когда число минут оказывается большим 60: определяется количество часов, содержащихся в поле `minutes` данного пользователя, и соответственно увеличивается количество часов в `hours`. Функция `int` возвращает целую часть своего аргумента (Perl обрабатывает и числа с плавающей запятой, поэтому необходимо использовать `int`).

Наконец, в строках 19–22 осуществляется вывод общего времени регистрации и количества входов в систему для каждого пользователя. Простая функция `print` выводит свои аргументы, как одноименная функция в `awk`. Обратите внимание, что вычисление переменных можно производить внутри оператора `print`, как сделано в строках 19 и 22. Однако, если вы хотите осуществлять форматированный вывод текста, нужно использовать функцию `printf` (аналогичную своему эквиваленту в C). В данном случае мы хотим установить минимальную ширину поля для вывода часов и минут в два символа и дополнить слева нулями недостающие позиции. Для этого используется команда `printf` в строке 21.

Если сохранить этот сценарий в файле `logintime`, то выполнить его можно следующей командой:

```
papaya$ last | logintime
User johnsonm, total login time 01:07, total logins 11.
User kibo, total login time 00:42, total logins 3.
User linus, total login time 98:50, total logins 208.
User mdw, total login time 153:03, total logins 290.
papaya$
```

Конечно, этот сценарий не вполне подходит в качестве учебника по языку Perl, но он должен дать вам представление о его возможностях. Мы советуем почитать одну из имеющихся прекрасных книг по Perl, чтобы узнать о нем больше.

Некоторые дополнительные возможности

В предыдущем примере мы познакомились с некоторыми часто используемыми в Perl функциями, представив пример живой рабочей программы. Однако в Perl много других функций, как популярных, так и менее известных.

Как уже говорилось, Perl обеспечивает механизм создания отчетов, выходящий за рамки стандартных функций `print` и `printf`. С помощью этого механизма программист задает «формат», описывающий вид каждой страницы отчета. Например, мы могли бы включить такое определение формата в нашу программу:

```
format STDOUT_TOP =
User           Total login time     Total logins
-------------- -------------------- -------------------
.
format STDOUT =
@<<<<<<<<<<<<< @<<<<<<<<            @####
$user,         $thetime,             $logins{$user}
.
```

Определение `STDOUT_TOP` описывает заголовок отчета, который будет печататься вверху каждой страницы вывода. Формат `STDOUT` описывает вид каждой выводимой строки. Описание каждого поля начинается с символа `@`, символы `@<<<<` указывают на выровненное по левому краю текстовое поле, а символы `@####` задают числовое поле. Строка под описаниями полей задает имена переменных, которые должны выводиться в этих полях. Мы использовали переменную `$thetime` для хранения отформатированной строки времени.

Чтобы использовать при выводе этот отчет, заменим строки 19–22 в исходном сценарии на следующие:

```
$thetime = sprintf("%02d:%02d", $hours{$user}, $minutes{$user});
write;
```

Первая строка использует функцию `sprintf` для форматирования строки времени и сохранения ее в переменной `$thetime`; вторая строка является командой `write`, сообщающей Perl о необходимости вывести строку с использованием установленного формата отчета.

Используя такой формат, мы должны получить нечто вроде следующего:

```
User           Total login time     Total logins
-------------- -------------------- -------------------
johnsonm       01:07                   11
kibo           00:42                    3
linus          98:50                  208
```

```
mdw             153:03                  290
```

С помощью других форматов отчета можно получить другие (и лучше выглядящие) результаты.

Perl поставляется с огромным количеством модулей, которые можно вставлять в программы, чтобы получать быстрый доступ к очень мощным функциям. В популярном электронном архиве CPAN (Comprehensive Perl Archive Network – полный сетевой архив Perl) содержится еще больше модулей: сетевые модули, позволяющие отправлять электронную почту и выполнять другие сетевые задачи, модули для дампа данных и отладки, модули для обработки дат и времени, модули для математических функций. Этот список можно продолжить на многие страницы.

Узнав об интересном модуле, проверьте сначала, не установлен ли он уже в вашей системе. Вы можете просмотреть каталоги, где размещаются модули (вероятно, в */usr/lib/perl5*), или попытаться загрузить модуль и посмотреть, как он работает. Например, команда

```
$ perl -MCGI -e 1
Can't locate CGI in @INC...
```

с прискорбием извещает, что модуль *CGI.pm* отсутствует в вашей системе. Модуль *CGI.pm* весьма популярен и включается в стандартный дистрибутив Perl, откуда его можно установить. Однако за многими другими модулями вам придется обратиться к CPAN (а некоторых нет и в CPAN). CPAN, сопровождаемый Ярко Хиетаньеми (Jarkko Hietaniemi) и Андреасом Кенигом (Andreas Konig), лежит на десятках зеркальных сайтов по всему миру, поскольку очень многие хотят загрузить его модули. Проще всего попасть в CPAN, посетив веб-страницу *http://www.perl.com/CPAN*.

Следующая программа, которую мы хотели сделать короткой и потому не стали искать практическую задачу, показывает два модуля, один из которых обрабатывает даты и время довольно сложным образом, а другой отправляет электронную почту. Недостатком использования таких мощных функций является необходимость загрузки для них огромного объема программного кода, что делает весьма большим размер программы при выполнении:

```
#! /usr/local/bin/perl

# Демонстрация работы модулей Date и Mail
use Date::Manip;
use Mail::Mailer;

# Иллюстрация использования модуля Date::Manip
if ( Date_IsWorkDay( "today", 1) ) {

    # Сегодня рабочий день
    $date = ParseDate( "today" );
}
else {
    # Сегодня нерабочий день, поэтому выберем очередной рабочий день
    $date=DateCalc( "today" , "+ 1 business day" );
}

# Преобразуем дату из компактной формы в читаемую "April 8"
$printable_date = UnixDate( $date , "%B %e" );
```

```
# Иллюстрация работы модуля Mail::Mailer
my ($to) = "the_person\@you_want_to.mail_to";
my ($from) = "owner_of_script\@system.name";

$mail = Mail::Mailer->new;

$mail->open(
            {
                From => $from,
                To => $to,
                Subject => "Автоматическое напоминание",
            }
           );
print $mail <<"MAIL_BODY";
Вы получите это почтовое сообщение,
если находитесь на работе
$printable_date,
или позже.
MAIL_BODY

$mail->close;

# Почта отправлена! (Предполагается, что не было ошибок.)
```

Пакеты очень просто использовать благодаря тому, что в версию Perl 5 добавлены функции объектно-ориентированного программирования. Модуль Date, использованный в предыдущем примере, не является объектно-ориентированным, в отличие от модуля Mail. Переменная `$mail` является объектом типа Mailer, упрощая работу почты благодаря таким методам, как `new`, `open` и `close`.

Для солидных задач, таких как синтаксический разбор HTML, нужно считать подходящий пакет CGI, использовать команду `new`, чтобы создать подходящий объект, и в вашем распоряжении окажутся все функции, необходимые для разбора HTML.

При желании снабдить свой Perl-сценарий графическим интерфейсом воспользуйтесь модулем Tk, который первоначально был разработан для применения с языком Tcl, или модулем Gtk, который использует более новый GIMP Toolkit (GTK), или модулем Qt, использующим Qt-инструментарий, являющийся основой и для KDE. В книге «Learning Perl/Tk» Нэнси Уолш (Nancy Walsh), O'Reilly, рассказано, как создавать графику с помощью этого модуля.

Другой скрытой возможностью Perl является его способность более или менее прямо осуществлять некоторые системные вызовы UNIX, в том числе для связи между процессами. Например, Perl предоставляет доступ к функциям `msgctl`, `msgget`, `msgsnd` и `msgrcv` из SystemV IPC. Perl поддерживает также реализацию сокетов BSD, что позволяет осуществлять коммуникации по TCP/IP непосредственно из программы на Perl. Теперь C не является единственным языком для сетевых демонов и клиентов. Программа на Perl с функциями IPC может быть очень мощной, особенно с учетом того, что многие реализации систем «клиент-сервер» нуждаются в развитых функциях обработки текста, таких как предоставляемые Perl. Обычно проще делать разбор команд протокола, которыми обмениваются клиент и сервер, в сценарии Perl, чем писать для этого сложную программу на C.

В качестве примера возьмем хорошо известный демон SMTP, управляющий отправкой и получением электронной почты. Протокол SMTP использует такие внутренние команды, как *recv from* и *mail to*, для предоставления клиенту возможности взаимодействия с сервером. Как клиент, так и сервер могут быть написаны на языке Perl и иметь полный доступ к имеющимся в Perl функциям обработки текстов и файлов, а также важным функциям связи через сокеты.

Perl прочно обосновался в CGI-программировании, то есть написании небольших программ, выполняемых на веб-сервере и делающих веб-страницы более интерактивными.

За и против

Одной из особенностей (некоторые говорят «проблем») Perl является его способность значительно сокращать – и делать непонятным – программный код. В предыдущем примере мы использовали несколько общепринятых сокращений. Например, входные данные в сценарии Perl считываются в переменную `$_`. Однако большинство операций производится с переменной `$_` по умолчанию, поэтому нет необходимости обращаться к ней по имени.

Perl часто дает возможность сделать одно и то же разными способами, что может оказаться благодатью или проклятием, в зависимости от того, как на это посмотреть. В книге «Programming Perl» Ларри Уолл приводит такой пример короткой программы, которая просто выводит то, что поступает на ее стандартный ввод. Все нижеследующие операторы эквивалентны:

```
while ($_ = <STDIN>) { print; }
while (<STDIN>) { print; }
for (;<STDIN>;) { print; }
print while $_ = <STDIN>;
print while <STDIN>;
```

Программист может применять наиболее подходящий в его ситуации синтаксис.

Perl популярен не только благодаря своим удобствам. Программистов привлекает его эксцентричность. Программисты на Perl постоянно соревнуются в том, кто напишет более замысловатый код. Perl идеально подходит для интересных клуджей, изящных хаков и как очень хорошего, так и очень плохого стилей программирования. Программисты в UNIX считают его сложным языком для работы. Даже если Perl на ваш вкус слишком причудлив, его все же можно оценить за артистичность. Возможность назвать себя «крутым программистом на Perl» (Perl hacker) служит предметом гордости в сообществе UNIX.

Java

Java – это объектно-ориентированный язык, поддерживающий работу в сети, разработанный Sun Microsystems. Java вызывает очень большой интерес в компьютерном сообществе, поскольку стремится предоставить безопасный язык для запуска приложений, загружаемых с сайтов World Wide Web. Идея проста: позволить веб-броузерам загружать апплеты Java, которые могут исполняться на машине клиента. Многие популярные броузеры, включая Mozilla и Firefox, основанный на том же движке броузер рабочего стола GNOME Galeon и веб-броузер

из KDE Konqueror (глава 5) – все они обладают поддержкой Java. Кроме того, Java Developer's Kit и другие инструментальные средства перенесены на Linux.

Но язык Java пригоден не только для создания апплетов. В последнее время он все чаще используется в качестве универсального языка программирования, не представляющего большой сложности для начинающих и часто используемого для программирования приложений в архитектуре «клиент-сервер» благодаря встроенным сетевым библиотекам. Ряд учебных заведений также выбрали Java для своих курсов программирования.

Перспективы Java

Все это может не вызвать у вас вдохновения. В конце концов, существует много объектно-ориентированных языков программирования, а с помощью подключаемых модулей Mozilla можно загружать с веб-серверов исполняемые модули и запускать их на локальной машине.

Но Java – это много больше, чем объектно-ориентированный язык программирования. Одной из самых впечатляющих его особенностей является *независимость от платформы*. Это означает, что можно написать на Java программу и скомпилировать ее, а затем установить почти на любой машине, будь то скромный 386-й компьютер под Linux, мощный Pentium IV с последней раздутой системой Microsoft или мейнфрейм IBM. Sun Microsystems называет это «Write Once, Run Anywhere», то есть «однажды написав, выполняешь всюду». К сожалению, действительность сложнее, чем цели проектирования. Существуют небольшие, но срывающие все планы различия, из-за которых программа работает на одной платформе и отказывается работать на другой. С появлением новой библиотеки GUI Swing в Java 2 был сделан большой шаг в направлении исправления этого положения.

Изящная возможность скомпилировать код один раз и иметь после этого возможность исполнять его на других машинах осуществляется благодаря виртуальной машине Java – Java Virtual Machine (JVM). Компилятор Java не создает объектный код для конкретного типа процессора и операционной системы, как это делает *gcc*, он генерирует байт-код для виртуальной машины JVM. Эта «машина» не существует в железе (пока), а является спецификацией. В спецификации сказано, какие так называемые коды операций (opcodes) понимает машина и что она делает, когда встречает их в объектном файле. Программа распространяется в двоичном виде, содержащем так называемые байт-коды, соответствующие спецификации JVM.

Теперь все, что вам нужно, – это программа, реализующая виртуальную машину Java на вашем конкретном компьютере и операционной системе. В данный момент они имеются практически для каждой платформы – ни один поставщик не может позволить себе не предоставить виртуальную машину Java вместе со своим оборудованием или операционной системой. Эта программа называется также *интерпретатором Java*, поскольку интерпретирует коды операций, скомпилированные для JVM, и транслирует их в машинный код для конкретного типа процессора.

Эта отличительная особенность, делающая Java одновременно компилируемым и интерпретируемым языком, позволяет написать, скомпилировать свою Java-

программу и передать ее кому-нибудь другому, и она сможет выполняться при наличии интерпретатора Java, независимо от аппаратной платформы и операционной системы.

Увы, за независимость Java от платформы приходится платить. Поскольку объектный код не является объектным кодом для реально существующего процессора, его необходимо дополнительно обработать, в результате чего написанные на Java программы выполняются в десять-двадцать раз медленнее, чем аналогичные программы, написанные, скажем, на языке C. В одних случаях это не имеет значения, а в других – просто недопустимо. Существуют так называемые *компиляторы just-in-time* (JIT – во время исполнения), которые сначала транслируют объектный код для виртуальной машины Java в родной объектный код, а затем его исполняют. Когда тот же объектный код нужно выполнить вторично, можно использовать прекомпилированный собственный код (native code) без всякой интерпретации, в результате чего выполнение происходит быстрее, но все же скорости C-программ не достижимы. Новейшие компиляторы широко используют технологию компиляции *just-in-time*, но пока скорость исполнения, «сравнимая со скоростью работы C-программ», остается недостижимой.

В Java также различают *приложения* (*applications*) и *апплеты* (*applets*). Приложения являются самостоятельными программами, которые можно запустить из командной строки или с рабочего стола, и они будут вести себя как обычные приложения. Напротив, апплеты являются программами (обычно меньшего размера), выполняемыми внутри веб-броузера. (Для работы этих программ в броузер должен быть встроен интерпретатор Java.) При просмотре веб-страницы, содержащей апплет Java, веб-сервер посылает вам объектный код апплета, и броузер его исполняет. Это может быть использовано для любых задач – от простой мультипликации до полных интерактивных банковских систем.[1]

Читая об апплетах Java, вы могли подумать: «А что если в апплете содержится злонамеренный код, который шпионит за данными на моем жестком диске, а возможно, даже уничтожает или портит файлы?» Разумеется, это было бы возможно, если бы разработчики Java не предусмотрели многоэтапные контрмеры против таких атак: все апплеты Java исполняются в так называемой «песочнице», предоставляющей им весьма ограниченный доступ к ресурсам компьютера. Например, апплеты Java могут выводить текст на экран, но не могут читать данные из локальной файловой системы и даже записывать в нее, не получив на это вашего явного разрешения. Хотя парадигма песочницы уменьшает полезность апплетов, но в то же время повышает степень защищенности ваших данных. В последних версиях Java можно установить требуемый уровень защиты и получить благодаря этому дополнительную гибкость. Следует упомянуть, что есть сведения о наличии серьезных брешей в системе безопасности, вызванных использованием Java в броузерах, хотя все те, которые обнаружены, исправлены в современных веб-броузерах.

---

[1] Один из авторов все свои финансовые операции с банком производит через апплет Java, поставляемый банковским сервером при просмотре определенного раздела веб-сервера банка.

Если вы заинтересовались Java, мы можем порекомендовать книгу Брюса Эккеля (Bruce Eckel) «Thinking in Java».[1] Она освещает большинство тем Java, которые нужно знать, а также учит основным принципам программирования. Среди других книг по Java, в которые стоит заглянуть, отметим «Learning Java» и «Core Java» Кэя Хорстманна.[2]

Как получить Java для Linux

К счастью, на Linux перенесен так называемый JDK (Java Developers Kit – комплект инструментальных средств разработки), поставляемый Sun Microsystems для Solaris и Windows и являющийся реализацией Java, на которую нужно ориентироваться. В прошлом существовал разрыв между появлением новой версии JDK для Solaris и Windows и наличием JDK для Linux. В настоящее время такой проблемы нет.

«Официальная» Java-реализация JDK содержит компилятор, интерпретатор и несколько сопутствующих программных средств. Существуют и другие комплекты инструментальных средств для Linux, нередко распространяемые с открытыми исходными текстами. Здесь мы расскажем о JDK, поскольку он является стандартом. Есть и другие реализации JDK для Linux, в том числе очень добротная, производства IBM. Возможно, одна из них есть в вашем дистрибутиве.

Следует отметить, что в большинстве дистрибутивов уже есть JDK для Linux, поэтому может оказаться проще установить тот комплект, который содержится в пакете. Однако JDK быстро развивается, поэтому может оказаться предпочтительнее установить более новый пакет, чем тот, который содержится в вашем дистрибутиве.

Все необходимое для работы с Java в Linux можно найти по адресу *http://www.sun.org*. Здесь можно найти документацию и новости, а также загрузить копию JDK для вашей машины.

После распаковки и установки JDK согласно имеющимся инструкциям в вашем распоряжении окажется несколько новых программ: *javac* – компилятор Java, *java* – интерпретатор и *appletviewer* – небольшая программа с графическим интерфейсом, позволяющая запускать апплеты без полномасштабного веб-броузера.

Python

В последнее время большое внимание уделяется языку Python, представляющему собой мощную смесь различных парадигм и стилей программирования. Например, это один из очень немногих интерпретирующих объектно-ориентированных языков программирования (еще одним примером интерпретирующего объектно-ориентированного языка может служить Perl, но это свое качество он обрел уже будучи в зрелом возрасте). Приверженцы языка Python утверждают,

---

[1] Брюс Эккель «Философия Java. Библиотека программиста», 3-е издание. – Пер. с англ. – СПб.: Питер, 2003.

[2] К. Хорстманн «Java 2. Библиотека профессионала, том 1. Основы», «Java 2. Библиотека профессионала, том 2. Тонкости программирования», 7-е издание. – Пер. с англ. – М.: Вильямс, 2006.

что он исключительно прост в изучении. Python был практически целиком спроектирован и написан Гвидо Ван Россумом (Guido van Rossum). Свое название язык получил благодаря тому, что во время работы над интерпретатором автор смотрел по британскому телевидению телешоу «Monty Phyton's Flying Cicus». Введение в язык можно найти в книге «Learning Python» (O'Reilly), а детальное описание – в книге «Programming Python».[1]

У языка Perl, каким бы прекрасным и удобным он ни был, есть один недостаток – по крайней мере, так считают многие: один и тот же программный код можно написать многими различными способами. С этим связано мнение о Perl как о языке, на котором просто писать, но который трудно читать. (Имеется в виду, что другой программист может сделать все не так, как вы, а вам будет трудно разобраться с его стилем.) Из этого следует, что Perl может оказаться не тем языком, на котором следует разрабатывать приложения, требующие сопровождения в течение последующих лет.

Если вы привыкли писать программы на C, C++ или Java, но иногда вам приходится писать сценарии, синтаксис Perl может показаться вам слишком непривычным – например, нужно вводить знак доллара перед именем переменной:

```
foreach $user ...
```

Прежде чем более подробно рассматривать, что представляет собой Python, выскажем предположение, что выбор между Perl и Python в значительной мере является вопросом «религии» – так же, как выбор между Emacs и *vi* или между KDE и GNOME. Perl и Python заполняют собой пробел между реальными языками, такими как C/C++/Java, и языками сценариев, которые встроены в командные оболочки *bash*, *zsh* или *tcsh*.

В противоположность Perl, Python с самого начала проектировался как реальный язык программирования, многие конструкции которого навеяны языком C. Из этого следует, что программы на языке Python читать проще, чем программы, написанные на Perl, хотя они получаются несколько длиннее.

Python – это объектно-ориентированный язык, но он не вынуждает программиста писать в объектно-ориентированном стиле, если он не испытывает в этом необходимости. Благодаря этому можно начать писать сценарий, не используя объектно-ориентированных технологий, но по мере продвижения вперед и увеличения размеров и сложности сценария легко преобразовать его для использования объектов.

Сценарии Python интерпретируются, то есть не требуется ждать конца долгого процесса компиляции. Программы Python динамически компилируются в байткод, благодаря чему они все же работают с приличной скоростью. В повседневной работе этого можно и не заметить, за исключением того, что если вы напишете файл с расширением *.py*, Python создаст файл с расширением *.pyc*.

В Python есть списки, кортежи, строки и ассоциативные массивы (или, как принято называть в Python, *«словари»* (*dictionaries*)), которые встроены в синтаксис языка, что очень упрощает работу с этими типами.

---

[1] Марк Лутц «Программирование на Phyton», 2-е издание. – Пер. с англ. – СПб: Символ-Плюс, 2002.

Python поставляется с обширной библиотекой, близкой по мощи к той, которую мы видели в Perl. Полный справочник по библиотеке см. по адресу *http://www.python.org/doc/current/lib/lib.html*.

Анализ результатов работы команды last на языке Python

В большинстве случаев объяснять принципы работы с языками программирования лучше всего на примерах, поэтому ниже вашему вниманию будет представлен небольшой сценарий на языке Python, который выполняет ту же работу, что и предшествующий сценарий Perl. Первое впечатление – этот сценарий существенно длиннее, чем сценарий Perl. Учтите, что тут мы заставляем Python «соревноваться» с Perl в той области, где последний наиболее силен. Компенсацией служит то, что этот сценарий, по нашему мнению, проще читать.

Обратите также внимание на отступы. В то время как в большинстве языков программирования отступы необязательны и просто улучшают читаемость кода, в Python они обязательны и составляют одну из его отличительных его особенностей.

```
1   #!/usr/bin/python
2
3   import sys, re, string
4
5   minutes = {}
6   count = {}
7   line = sys.stdin.readline()
8   while line:
9     match = re.match( "^(\S*)\s*.*\(([0-9]+):([0-9]+)\)\s*$", line )
10    if match:
11      user = match.group(1)
12      time = string.atoi(match.group(2))*60 + string.atoi(match.group(3))
13      if not count.has_key( user ):
14        minutes[ user ] = 0
15        count[ user ] = 0
16      minutes[ user ] += time
17      count[user] += 1
18    line = sys.stdin.readline()
19
20    for user in count.keys():
21      hour = `minutes[user]/60`
22      min = minutes[user] % 60
23      if min < 10:
24        minute = "0" + `min`
25      else:
26        minute = `min`
27      print "User " + user + ", total login time " + \
28            hour + ":" + minute + \
29            ", total logins " + `count[user]`
```

Этот сценарий должен быть самоочевиден, за некоторыми исключениями. В строке 3 импортируются необходимые для работы библиотеки. Например, импортировав `string`, мы можем в строке 12 воспользоваться этой библиотекой, применив входящий в нее метод `atoi`.

В строках 5 и 6 инициализируются два словаря. В отличие от Perl, мы должны инициализировать их, прежде чем присваивать им значения. Строка 7 читает строку из стандартного ввода. Когда строк для чтения больше нет, метод `readline` возвращает значение `None`, что эквивалентно пустому указателю.

Строка 9 сравнивает строку, прочтенную из `stdin`, с регулярным выражением и возвращает объект соответствия. У этого объекта есть метод для доступа к отдельным элементам совпадения. В строке 21 результат деления `minutes[user]/60` преобразуется в строку. Это осуществляется с помощью двух обратных апострофов.

Разработка программы-калькулятора на языке Python

В этом разделе приводится пример более сложного приложения, в котором используются классы. Данное приложение является реализацией калькулятора, выполняющего вычисления с использованием обратной польской нотации. Внешний вид приложения приводится на рис. 21.1. Для построения графического интерфейса была использована библиотека Qt, которая представляет собой библиотеку классов C++ и имеет обертки для многих других языков программирования, таких как Perl, Python и Java.

Программа состоит из двух классов: `Display`, который отвечает за отображения области вывода чисел (дисплея калькулятора), и `Calculator`, который собственно и реализует функциональность калькулятора:

```
 1 #!/usr/bin/python
 2 import sys, string
 3 from qt import *
 4
 5 class Display(QTextEdit):
 6   def __init__( self, parent ):
 7     QTextEdit.__init__( self, parent )
 8     self.setAlignment( Qt.AlignRight )
 9     self.setReadOnly( 1 )
10
11   def pop( self ):
12     lines = self.paragraphs( )
13     if lines == 0:
```

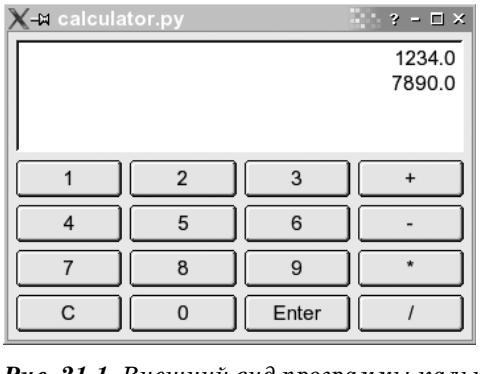


Рис. 21.1. *Внешний вид программы-калькулятора, написанной на языке Python*

```
      return 0

    res = QString.stripWhiteSpace(self.text( lines-1 ))
    self.removeParagraph(lines-1)

    if ( res.isEmpty( ) ):
      return 0

    return res.toFloat( )[0]

  def push( self, txt ):
    self.append( `txt` )

  class Calculator(QDialog):
    # Конструктор
    def __init__(self, parent):
      QDialog.__init__(self, parent)
      vlay = QVBoxLayout( self, 6 )
      self.insertMode = 0

    # Создать дисплей
    self.edit = Display( self )
    vlay.addWidget( self.edit )

    # Создать клавиатуру
    index = 0
    for txt in [ "1", "2", "3", "+", "4", "5", "6", "-", "7", "8", "9", \
                 "*", "C", "0", "Enter", "/" ]:
      if (index%4) == 0:
        hlay = QHBoxLayout( vlay )
      index = index+1

      but = QPushButton( txt, self )
      but.setAutoDefault(0)
      QObject.connect( but, SIGNAL( "clicked( )" ), self.buttonClicked )
      hlay.addWidget( but )

  # Функция обработки нажатия клавиши
  def buttonClicked(self):
    txt = self.sender( ).text( ) # Надпись на нажатой кнопке.
    if txt == "Enter":
      self.insertMode = 0

    elif txt in [ "+", "-", "*", "/" ]:
      val1 = self.edit.pop( )
      val2 = self.edit.pop( )
      self.edit.push( self.evaluate( val2, val1, txt ) )
      self.insertMode = 0

    elif txt == "C":
      self.edit.pop( )
      self.insertMode = 0

    else: # Нажата кнопка с цифрой.
      if self.insertMode:
```

```
69          val = self.edit.pop( )
70        else:
71          self.insertMode = 1
72          val = 0
73        val = val*10+ txt.toFloat( )[0]
74        self.edit.push(val)
75
76    def evaluate( self, arg1, arg2, op ):
77      if ( op == "+" ):
78        return arg1 + arg2
79      elif ( op == "-" ):
80        return arg1 - arg2
81      elif ( op == "*" ):
82        return arg1 * arg2
83      elif (op == "/" ):
84        return arg1 / arg2
85
86 # Основная программа
87 app=QApplication(sys.argv)
88 cal = Calculator(None)
89 cal.show( )
90 app.exec_loop( )
```

Поначалу эта программа может показаться достаточно объемной, с другой стороны, в 90 строк программного кода удалось вместить полноценное приложение с графическим интерфейсом. На самом деле большую часть работы взяла на себя библиотека Qt, но поскольку в этой книге мы не рассматриваем вопросы программирования с использованием библиотеки Qt, мы не будем углубляться в эти проблемы слишком глубоко. Рассмотрим программу по частям.

Начнем со строк, откуда начинается исполнение программы, а именно со строк 86–90. Конечно же, исполнение программы начинается с первой строки, но первые 85 строк являются простыми определениями классов, интерпретация которых не приводит к выполнению каких-либо действий.

В строке 87 создается экземпляр класса `QApplication`: в отличие от других языков программирования, таких как Java или C++, здесь для создания экземпляра класса не используется ключевое слово `new` – достаточно обратиться к классу по имени. Определение класса `QApplication` находится в *qt.py*, который особым образом подключается к программе в строке 3: данная строка говорит, что все имена из этого модуля должны быть включены в пространство имен программы, благодаря этому удалось избежать необходимости писать `qt.QApplication`. Такой подход не всегда можно приветствовать, однако в нашем случае он вполне оправдан, поскольку имена всех классов библиотеки Qt уже как бы находятся в *отдельном пространстве имен* благодаря тому, что они начинаются с символа `Q`.

Экземпляр `QApplication`, созданный в строке 87, – это волшебный объект, который занимается обработкой всех событий, возникающих в недрах Qt. Обработка событий запускается вызовом метода `enter_loop()` в строке 90. Более подробное описание того, что происходит внутри, далеко выходит за рамки данной книги, а кроме того, оно не нужно, чтобы понять пример.

В строке 88 создается экземпляр класса `Calculator`, а в строке 89 вызывается его метод `show()`, который выполняет отображение диалога на экране.

Теперь взглянем на класс `Display` в строках 5–25. Этот класс наследует свойства и методы класса `QTextEdit`, который представляет собой элемент графического интерфейса, предназначенный для редактирования многострочного текста. Наследование выполнено в строке 5, где в круглых скобках указано имя суперкласса. Множественное наследование также возможно, в этом случае в круглых скобках следует указать список суперклассов, разделенных запятыми.

Строка 6 – это конструктор класса. Конструктор неявно вызывается всякий раз, когда создается экземпляр класса. Первый аргумент – ссылка на сам экземпляр. Эта ссылка необходима всякий раз, когда потребуется обратиться к методу экземпляра.[1]

В строке 7 вызывается конструктор суперкласса класса. Это необходимо, чтобы иметь возможность передать суперклассу ссылку на родительский класс.[2] Ссылка на родительский класс, помимо всего прочего, необходима для корректной работы графического интерфейса.

В строках 8 и 9 вызываются методы самого объекта. Эти методы определены в суперклассе, но они доступны самому классу-наследнику.[3]

В строках 11–22 находится определение метода `pop()`. Обратите внимание: ссылка на сам объект передается методу в виде первого аргумента. При обращении к такому методу ссылка на объект не передается в первом аргументе – эту работу берет на себя сам язык Python. Такой вызов можно наблюдать в строке 58.

Реализация методов `pop()` и `push()` связана главным образом с обслуживанием библиотеки Qt, поэтому мы не будем касаться всех тонкостей их работы и лишь в общих чертах опишем, что они делают. Метод `push()` добавляет число в конец поля редактирования, тогда как `pop()` выполняет обратное действие – считывает последнюю строку из поля редактирования, преобразует ее в число и возвращает это число вызывающей программе.

---

[1] В таких языках, как C++ или Java, нет необходимости явно указывать объект при вызове методов из функций-членов. Однако это необходимо делать в Python, где нельзя заменить строку 8 строкой `setAlignment(Qt.AlignRight)`.

[2] В терминологии объектно-ориентированного программирования понятия «суперкласс» и «родительский класс» суть одно и то же – это класс, стоящий выше в иерархии наследования. Однако в терминологии библиотеки Qt (по крайней мере, в той ее части, которая касается элементов графического интерфейса) под *родительским* (*parent*) классом (если быть точнее, это даже не класс, а экземпляр класса – виджет) понимается класс, владеющий окном или его участком, в пределах которого отображается данный конкретный экземпляр графического элемента. Например, класс `QApplication` не является родительским (в смысле наследования) для класса `Qlabel`, но может предоставлять свое окно для отображения текста метки, и потому в терминологии Qt называется «parent» (родительский). В русскоязычной литературе такие взаимоотношения между классами принято называть «владелец – подчиненный». В данном тексте под термином «суперкласс» понимается класс, стоящий выше в иерархии наследования, а под термином «родительский» – класс-владелец, предоставляющий область экрана, которой он владеет, подчиненным элементам для отображения. – *Примеч. перев.*

[3] Все методы классов в языке Python являются виртуальными, как и в языке Java, в отличие от C++.

Теперь перенесем внимание на класс `Calculator`, который реализует графический интерфейс всего приложения. Этот класс является диалогом и потому наследует класс `QDialog`, как это видно в строке 27.

И снова основная часть программного кода занимается обслуживанием библиотеки Qt, что не представляет для нас интереса в данном контексте. Хотя здесь есть один интересный момент, с которым мы еще не сталкивались, – переменные экземпляра класса. В строках 32, 55, 61, 65 и 71 выполняется присваивание значения переменной `self.insertMode`, и в строке 68 выполняется чтение этой переменной. Вследствие использования части `self.` перед именем переменной `self.insertMode` становится локальной по отношению к объекту – это называется *«переменная экземпляра класса»*. Если бы в программе имелось несколько экземпляров класса `Calculator`, каждый из них обладал бы собственной копией переменной `insertMode`. В отличие от таких языков программирования, как C++ или Java, в языке Python переменные экземпляра класса могут не объявляться заранее. Первая же попытка присваивания значения такой переменной создаст ее точно так же, как это делается в случае с локальными переменными.

Все необходимые сведения о Python можно найти на сайте *http://www.python.org* или в книгах «Learning Python» и «Programming Python». Если вас заинтересовали вопросы программирования с использованием библиотеки Qt, тогда интересно будет почитать книгу «Programming with Qt» (O'Reilly). Кроме того, есть еще одна книга, посвященная вопросам разработки Qt-программ на языке Python: «Gui Programming With Python: Using the Qt Toolkit» (Opendocs).

Другие языки программирования

Существует много других популярных (и не очень популярных) языков для Linux. По большей части они работают в Linux так же, как в других UNIX-системах, поэтому много нового здесь не скажешь. Кроме того, их так много, что особенно подробно мы их не сможем осветить. Однако мы все же хотим проинформировать вас о том, что есть в наличии, и рассказать о некоторых различиях между языками и компиляторами.

Одна из недавних разработок в области языков сценариев – язык Ruby, созданный в Японии и нашедший там большое число поклонников. Это объектно-ориентированный язык сценариев, который еще шире, чем Python (если это вообще возможно), использует объекты.

Tcl (Tool Command Language) – язык, предназначавшийся в качестве связующего между различными программами и прославившийся главным образом своим простым инструментарием создания оконного интерфейса, Tk.

LISP является интерпретируемым языком, используемым во многих приложениях – от искусственного интеллекта до статистики. Он применяется преимущественно в компьютерных исследованиях, поскольку определяет ясный логический интерфейс для работы с алгоритмами. (В нем также используется множество скобок, которые так любят теоретики вычислительной техники.) Это функциональный и очень обобщенный язык программирования. Многие операции определяются в терминах рекурсии, а не линейных циклов. Выражения являются иерархическими, а данные представляются списками элементов.

Существует несколько интерпретаторов LISP для Linux. Emacs LISP является довольно полной реализацией языка с рядом возможностей, позволяющих непосредственно взаимодействовать с Emacs, например осуществлять ввод и вывод через буферы Emacs, но он может также использоваться и для приложений, не связанных с Emacs.

Следует упомянуть также CLISP – реализацию Common LISP Бруно Хайбле (Bruno Haible) из университета Карлсруэ и Микаэля Штолля (Michael Stoll) из университета Мюнхена. В него входят интерпретатор, компилятор и подмножество CLOS (Common LISP Object System – объектно-ориентированное расширение LISP). Существует также CLX – интерфейс Common LISP к X Window System, который работает под CLISP. CLX позволяет писать на LISP приложения для X. Austin Kyoto Common LISP – еще одна реализация LISP, которая также совместима с CLX.

Существует еще SWI-Prolog – полная реализация Prolog Яна Вилемакера (Jan Wielemaker) из университета Амстердама. Prolog – это основанный на логике язык, позволяющий делать логические утверждения, определять эвристики для проверки этих утверждений и принимать основанные на них решения. Этот язык удобен для создания приложений искусственного интеллекта.

Имеется также несколько интерпретаторов Scheme, из которых следует упомянуть MIT Scheme – полный интерпретатор Scheme, отвечающий стандарту R[4]. Scheme – это диалект языка LISP, предлагающий более гладкую и общую модель программирования. Это хороший диалект LISP для применения в теоретической вычислительной технике и изучения алгоритмов.

Существуют по крайней мере две реализации Ada: AdaEd – интерпретатор Ada и GNAT – GNU Ada Translator. GNAT – развитый оптимизирующий компилятор Ada, который для Ada является тем же, что *gcc* для C и C++.

Кроме того, имеется еще два популярных транслятора для Linux: *p2c* – транслятор с Pascal в C и *f2c* – транслятор с FORTRAN в C. Если вы сомневаетесь, что эти два транслятора могут действовать как настоящие компиляторы, то напрасно. Как *p2c*, так и *f2c* показали себя зрелыми продуктами, готовыми для серьезного применения с Pascal и FORTRAN.

Существует по крайней мере один компилятор языка Object Pascal для Linux, который в состоянии компилировать некоторые программы, написанные в среде разработки Delphi. И наконец, имеется Kylix – версия среды разработки Delphi для Linux.

f2c совместим с FORTRAN-77, а также имеет ряд вспомогательных утилит. *ftnchek* является программой контроля FORTRAN, аналогичной *lint*. С помощью *f2c* на Linux были перенесены библиотека численных методов LAPACK и библиотека FORTRAN многократно увеличенной точности – *mpfun*. *toolpack* является коллекцией утилит FORTRAN, таких как средство печати исходных текстов, преобразователь точности и программа проверки переносимости.

В рамках проекта Mono были разработаны компилятор языка C# и среда времени исполнения как часть проекта GNOME, что сделало возможным программирование в стиле .NET в среде Linux.

Среди других разнообразных языков, доступных в Linux, можно указать интерпретаторы APL, Rexx, Forth, ML, Eiffel и транслятор Simula-to-C. GNU-версии средств компиляции *lex* и *yacc* (переименованные в *flex* и *bison*, соответственно),

используемые во многих программных пакетах, также перенесены на Linux. *lex* и *yacc* идеально подходят для создания разного рода синтаксических анализаторов или трансляторов и чаще всего используются при написании компиляторов.

Введение в программирование с использованием OpenGL

Прежде чем завершить эту главу обзором интегрированных сред разработки, в частности KDevelop, рассмотрим еще одну интереснейшую тему – программирование трехмерной графики с использованием библиотек OpenGL!

Конечно же, в этой книге мы не в состоянии охватить вопросы программирования для OpenGL должным образом, поэтому мы остановимся на простом примере и покажем, с чего начать и как OpenGL интегрируется с двумя наиболее популярными инструментальными средствами.

GLUT

Пакет GL Utility Toolkit был написан Марком Килгардом (Mark Kilgard) – знаменитостью в SGI. Пакет не является свободно распространяемым программным обеспечением, но поставляется с полным комплектом исходных текстов, причем совершенно бесплатно. Сила GLUT в его простоте, поскольку он специально предназначался для тех, кто только начинает изучать OpenGL. Копия GLUT поставляется в составе библиотеки Mesa, кроме того, существует свободно распространяемая реализация GLUT, доступная на сайте *http://freeglut.sourceforge.net/*. GLUT берет на себя всю черновую работу, как, например, создание окна и тому подобное, благодаря чему вы быстро сможете приступить к самому интересному – созданию программного кода OpenGL.

Чтобы воспользоваться возможностями пакета GLUT, сначала необходимо получить доступ к его определениям:

```
#include <GL/glut.h>
```

Далее в функции `main()` вызвать несколько процедур инициализации:

```
glutInit(&argc, argv)
```

чтобы инициализировать GLUT и дать ему возможность обработать параметры командной строки, а затем

```
glutInitDisplayMode( unsigned int mode )
```

где `mode` – битовая маска, содержащая значения констант из *glut.h*, объединенных по ИЛИ (OR). Мы будем использовать значение `GLUT_RGBA | GLUT_SINGLE`, что дает полноцветный режим отображения окна без двойной буферизации.

Размер окна устанавливается с помощью вызова

```
glutInitWindowSize(500,500)
```

и наконец создается окно функцией

```
glutCreateWindow("Some title")
```

Чтобы содержимое окна перерисовывалось, когда этого потребует оконная система, необходимо зарегистрировать функцию обратного вызова. Мы будем регистрировать функцию `disp` следующим образом:

```
glutDisplayFunc(disp)
```

Функция `disp()` – это то место, откуда будут производиться все обращения к библиотеке OpenGL. Начинается она с настройки механизмов трансформации объекта. Библиотека OpenGL использует для трансформации целый ряд матриц, одна из которых может быть назначена текущей вызовом функции `glMatrixMode(GLenum mode)`. Начальной является матрица `GL_MODELVIEW`, которая описывает трансформацию трехмерных объектов перед проецированием на экран. В нашем примере будет загружаться матрица тождественного преобразования с некоторым изменением масштаба и угла поворота.

Затем экран очищается и настраивается перо белого цвета шириной в четыре пиксела. Далее следуют операции рисования геометрической фигуры. Операции рисования в OpenGL обрамляются функциональными скобками `glBegin()` и `glEnd()`. Аргумент функции `glBegin()` управляет режимом интерпретации выполняемых операций.

Нам нужно нарисовать простой параллелепипед, поэтому сначала будут нарисованы четыре боковых ребра параллелепипеда, а затем два торцевых прямоугольника (в режиме `GL_LINE_LOOP`). Завершается процесс рисования вызовом функции `glFlush()`, которая выталкивает содержимое буфера рисования OpenGL на экран.

Чтобы сделать пример еще интереснее, в него была добавлена функция обратного вызова `timeout()`, которая вызывается по таймеру через каждые 50 миллисекунд. Эта функция каждый раз изменяет угол поворота фигуры и снова выводит ее на экран.

Ниже приводится полный исходный текст примера:

```
#include <GL/glut.h>

static int glutwin;
static float rot = 0.;

static void disp(void)
{
    float scale=0.5;
    /* трансформация сцены */
    glLoadIdentity( );
    glScalef( scale, scale, scale );
    glRotatef( rot, 1.0, 0.0, 0.0 );
    glRotatef( rot, 0.0, 1.0, 0.0 );
    glRotatef( rot, 0.0, 0.0, 1.0 );

    /* очистка экрана */
    glClear(GL_COLOR_BUFFER_BIT);

    /* нарисовать фигуру */
    glLineWidth( 3.0 );
    glColor3f( 1., 1., 1. );

    glBegin( GL_LINES ); /* длинные ребра параллелепипеда */
    glVertex3f( 1.0, 0.6, -0.4 ); glVertex3f( 1.0, 0.6, 0.4 );
```

```
    glVertex3f( 1.0, -0.6, -0.4 ); glVertex3f( 1.0, -0.6, 0.4 );
    glVertex3f( -1.0, -0.6, -0.4 ); glVertex3f( -1.0, -0.6, 0.4 );
    glVertex3f( -1.0, 0.6, -0.4 ); glVertex3f( -1.0, 0.6, 0.4 );
    glEnd( );

    glBegin( GL_LINE_LOOP ); /* торцевой прямоугольник */
    glVertex3f( 1.0, 0.6, -0.4 ); glVertex3f( 1.0, -0.6, -0.4 );
    glVertex3f( -1.0, -0.6, -0.4 ); glVertex3f( -1.0, 0.6, -0.4 );
    glEnd( );

    glBegin( GL_LINE_LOOP ); /* второй торцевой прямоугольник */
    glVertex3f( 1.0, 0.6, 0.4 ); glVertex3f( 1.0, -0.6, 0.4 );
    glVertex3f( -1.0, -0.6, 0.4 ); glVertex3f( -1.0, 0.6, 0.4 );
    glEnd( );

    glFlush( );
}

static void timeout( int value )
{
    rot++; if( rot >= 360. ) rot = 0.;
    glutPostRedisplay( );
    glutTimerFunc( 50, timeout, 0 );
}

int main( int argc, char** argv )
{
    /* инициализировать glut */
    glutInit(&argc, argv);

    /* установить режим отображения */
    glutInitDisplayMode(GLUT_RGBA | GLUT_SINGLE);

    /* размер окна */
    glutInitWindowSize(500,500);
    glutwin = glutCreateWindow("Running Linux 3D Demo");
    glutDisplayFunc(disp);

    /* определить цвет, который будет использоваться при очистке экрана */
    glClearColor(0.,0.,0.,0.);

    /* таймер анимации */
    glutTimerFunc( 0, timeout, 0 );

    /* вход в главный цикл программы */
    glutMainLoop( );
    return 0;
}
```

Qt

С целью демонстрации того, как можно программировать для OpenGL с использованием более универсальных средств разработки графического интерфейса, мы перепишем предыдущий пример на языке C++ с использованием библиотеки Qt. Библиотека Qt доступна на сайте *http://www.trolltech.com/* на условиях лицензии GPL и используется многими крупными проектами свободного программного обеспечения, такими как KDE.

Начинается пример с описания дочернего класса, наследующего класс QGLWidget, который является основным классом поддержки OpenGL в Qt. Класс QGLWidget работает аналогично любому другому классу, наследнику QWidget, с тем отличием, что рисование производится средствами OpenGL, а не QPainter. Роль функции обратного вызова из предыдущего примера теперь будет выполнять наша реализация виртуального метода paintGL(), но сам порядок работы с графикой останется без изменений. Пакет GLUT сам заботится об изменении области отображения при изменении размеров окна, но при работе с библиотекой Qt это придется делать нам самим. Для этого необходимо перекрыть виртуальный метод resizeGL(int w, int h). В данном примере мы просто будем вызывать функцию glViewport() и передавать ей новые размеры.

Анимация реализована с помощью класса QTimer, который подключается к методу timeout() и будет вызывать его каждые 50 миллисекунд. Метод updateGL() выполняет ту же роль, что и функция glutPostRedisplay() в пакете GLUT, – она заставляет приложение перерисовать экран.

Собственно команды рисования были опущены в примере, поскольку они нисколько не изменились. Ниже приводится полный исходный текст:

```
#include <qapplication.h>
#include <qtimer.h>
#include <qgl.h>

class RLDemoGLWidget : public QGLWidget {
    Q_OBJECT
public:
    RLDemoGLWidget(QWidget* parent,const char* name = 0);
public slots:
    void timeout( );
protected:
    virtual void resizeGL(int w, int h);
    virtual void paintGL( );
private:
    float rot;
};
RLDemoGLWidget::RLDemoGLWidget(QWidget* parent, const char* name)
    : QGLWidget(parent,name), rot(0.)
{
    QTimer* t = new QTimer( this );
    t->start( 50 );
    connect( t, SIGNAL( timeout( ) ),
             this, SLOT( timeout( ) ) );
}

void RLDemoGLWidget::resizeGL(int w, int h)
{
    /* установить новые размеры области отображения */
    glViewport(0, 0, (GLint)w, (GLint)h);
}

void RLDemoGLWidget::paintGL( )
{
    /* тело этого метода в точности повторяет содержимое */
    /* функции disp( ) из предыдущего примера */
```

```
    ...
}

void RLDemoGLWidget::timeout( )
{
    rot++; if( rot >= 360. ) rot = 0.;
    updateGL( );
}

int main( int argc, char** argv )
{
    /* инициализация Qt */
    QApplication app(argc, argv);

    /* создать виджет gl */
    RLDemoGLWidget w(0);
    app.setMainWidget(&w);
    w.resize(500,500);
    w.show( );
    return app.exec( );
}
#include "main.moc"
```

Интегрированные среды разработки

Хотя разработка программного обеспечения в UNIX (а следовательно, и в Linux) традиционно основана на командной строке, на других платформах разработчики привыкли к так называемым *интегрированным средам разработки* (Integrated Development Environment, IDE), в которых в виде одного приложения объединены редактор, компилятор, отладчик и, возможно, другие инструменты разработчика. Разработчики, имевшие опыт работы с такой средой, обычно бывают ошеломлены, когда сталкиваются с командной строкой Linux, в которой нужно ввести команду *gcc*.[1]

Чтобы угодить таким разработчикам-мигрантам, а также потому, что разработчики Linux требуют все больших и больших удобств, для Linux тоже были разработаны интегрированные среды. Их существует немало, но только одна из них – KDevelop – получила широкое распространение среди программистов, пишущих программы на C/C++. Еще одна среда разработки – Eclipse – весьма популярна в среде разработчиков, использующих язык Java.

KDevelop входит в состав проекта KDE, но может работать независимо от рабочего стола KDE. Она ведет учет всех файлов, входящих в ваш проект, генерирует файлы проектов (make-файлы), производит синтаксический анализ классов C++, в нее входят интегрированный отладчик и мастер, с помощью которого можно начать разработку нового приложения. Среда разработки KDevelop первоначально предназначалась в помощь при создании приложений для KDE, но может использоваться для разработки любых других программ, например обычных программ командной строки и даже приложений GNOME.

---

[1] Авторам непонятно, почему ввести команду *gcc* труднее, чем выбрать пункт в меню, но это может быть связано с нашим социальным статусом.

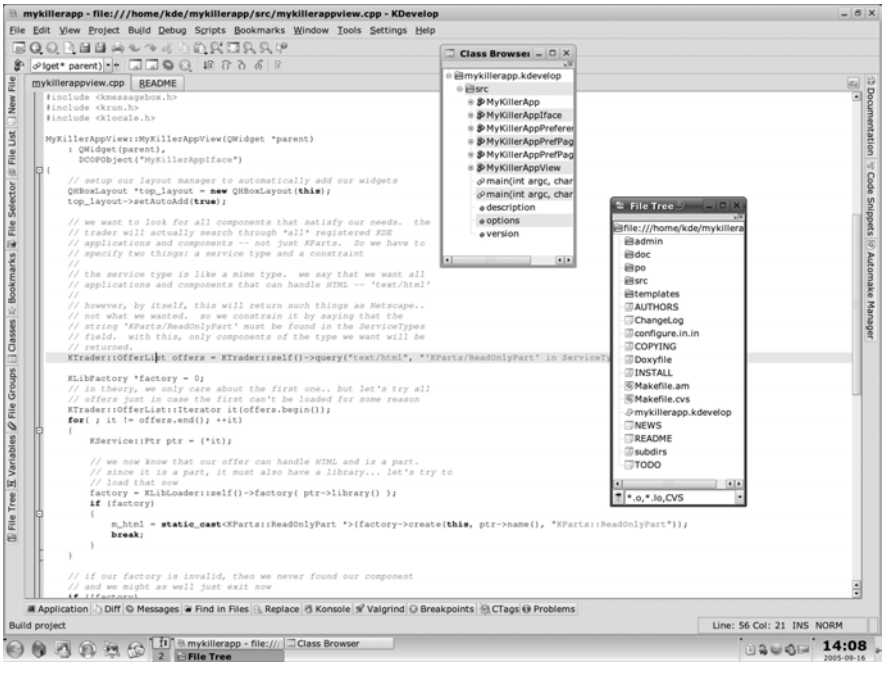


Рис. 21.2. *Интегрированная среда разработки KDevelop*

KDevelop слишком велика и богата функциями, чтобы знакомиться с ней в этой книге. Возбудим ваш аппетит хотя бы снимком экрана (рис. 21.2) и направим на сайт *http://www.kdevelop.org*, откуда можно выполнить загрузку и получить всю информацию, включая полную документацию.

Между прочим, Emacs и XEmacs представляют собой отличную IDE, в которую интегрированы многие дополнительные средства, например *gdb*, о чем говорилось выше в этой главе.

IV

Сетевые службы

В этой части книги вашему вниманию будут представлены различные аспекты, которые делают Linux мощной и весьма популярной системой. Среди них мы рассмотрим такие инструментальные средства, как веб-сервер, инструменты доставки электронной почты, базы данных и еще целый ряд других особенностей, представляющих самые передовые технологии в мире операционных систем. Здесь также будут продемонстрированы некоторые приемы защиты системы от потенциальных опасностей.

22

Запуск веб-сервера

В главе 13 мы подключились к сети. Возможно, это была тяжелая работа, но результат того стоил: теперь ваша система является частью сообщества. Если вы подключены к Интернету, следующим шагом будет получение доступа ко всем его богатствам.

Как в локальных сетях отдельных организаций, так и в Интернете, по общему мнению, самой привлекательной стороной является World Wide Web. В главах 3 и 5 мы уже рассматривали броузеры. Одна из самых захватывающих особенностей Linux – это возможность настроить свой собственный веб-сервер, именно это и станет темой данной главы.

Удобства, которые дает веб-сервер, работающий в системе, просто неоценимы. Мало того что вы сможете предоставить для всеобщего обозрения документацию или информацию из баз данных тем способом, который будет доступен пользователям любых систем, связанных с вами посредством сети, но вы еще сможете использовать целый ряд других инструментов (например, для администрирования системы), которые позволят выполнять административные задачи удаленно.

Однако, как только в системе появляется какой-либо сервер, приходится особое внимание уделять вопросам безопасности, потому что незначительные ошибки в настройках могут позволить злоумышленникам получить полный доступ к документам – едва ли кто-то захочет, чтобы веб-страницы оказались стерты или были уничтожены какие-либо данные. Внимательно ознакомьтесь с главой 26, прежде чем предоставлять доступ к вашему веб-серверу.

Настройка собственного веб-сервера

Настройка веб-сервера состоит из двух задач: настройки демона *httpd* и написания документов, которые вы собираетесь выложить на сервере. В этой книге мы не будем рассказывать об основах языка разметки HTML, потому что эти сведения широко распространены и, кроме того, существует масса инструментов с графическим интерфейсом, которые помогут вам в этом. Однако в главе 25 мы обсудим некоторые вопросы создания динамических веб-страниц (на основе информации, извлекаемой из баз данных).

httpd – это демон, обслуживающий HTTP-запросы на вашей машине. Обращение к любому документу по URL *HTTP* обслуживается *httpd*. Точно так же URL, начинающийся с *FTP*, обслуживается демоном *ftpd*, URL Goopher обслуживается *gopherd* и т. д. Нет единого веб-демона: каждый тип URL использует свой демон для доступа к информации на сервере.

Существуют несколько серверов HTTP. Здесь мы рассмотрим сервер Apache, отличающийся гибкостью и простотой настройки. В настоящее время имеются две версии Apache HTTP: 1.3 – более старая и широко распространенная версия, и 2.x – обладающая рядом характеристик специально для высокопроизводительных сайтов. Рекомендации по настройке, которые приводятся ниже, в равной степени пригодны для обеих версий.

Во всех современных версиях Linux сервером *httpd* по умолчанию должен быть Apache. Но если вы выбрали минимальную или настольную конфигурацию во время установки системы, может потребоваться установить Apache вручную. Возможно, вы захотите установить более новую версию, чем та, что входит в ваш дистрибутив, например более новую версию, обладающую более высокой степенью защищенности. Тогда можно загрузить двоичные файлы или исходные тексты с *http://httpd.apache.org* и собрать Apache самостоятельно. Веб-сайт *httpd.apache.org* содержит полную документацию программы.

Книга «Apache: The Definitive Guide» Бена Лурье (Ben Laurie) и Питера Лурье (Peter Laurie) охватывает все тонкости работы с Apache.

Местонахождение разных файлов Apache зависит от дистрибутива или от установленного вами пакета. Далее мы рассмотрим обычную установку. Для начала найдите различные части вашей системы:

/usr/sbin/httpd

Исполняемый файл, который, собственно, и является сервером. В дистрибутиве Debian он находится в каталоге */usr/sbin/apache*.

/etc/httpd

Содержит файлы с настройками *httpd*, наиболее важный из которых – *httpd.conf*. Позже мы обсудим, как вносить изменения в эти файлы. В системах Debian вместо этого каталога используется каталог */etc/apache*.

/usr/local/httpd

Содержит файлы HTML для обслуживания посетителей сайта. Этот каталог доступен всем пользователям Web вместе со всем, что в нем находится, и поэтому представляет серьезную угрозу безопасности для всего, что не является публичными данными.

/var/log/httpd

Содержит файлы журналов сервера.

Теперь наша задача – изменить файлы с настройками в соответствующем каталоге. В нем должны быть, по крайней мере, следующие четыре файла: *access.conf-dist*, *httpd.conf-dist*, *mime.types* и *srm.conf-dist*. (В дистрибутивах Apache 1.3 последних версий вместо окончания *-dist* используется расширение *.default*, а в Apache 2.x перед расширением файла добавлен фрагмент *-std*.) Скопируйте файлы с именами, заканчивающимися на *-dist*, и измените их для ва-

шей системы. Например, *access.conf-dist* надо скопировать в *access.conf* и отредактировать.

Последние версии Apache значительное число настроек выполняют автоматически, но мы расскажем, как вручную произвести настройку, если возникнут неполадки.

На сайте *http://httpd.apache.org* вы найдете полную документацию по настройке *httpd*. Здесь мы предлагаем примеры файлов с настройками, соответствующие реально работающему *httpd*.

httpd.conf

Файл *httpd.conf* является основным файлом, описывающим настройки сервера. Сначала скопируйте *httpd.conf-dist* в *httpd.conf* и отредактируйте его. Мы остановимся только на некоторых наиболее важных параметрах. Файл *httpd.conf-dist* снабжен весьма объемными комментариями.

Директива `ServerType` задает режим, в котором запускается сервер, – как автономный демон (в данном примере) или с помощью *inetd*. По многим причинам обычно лучше запускать *httpd* в автономном режиме. В противном случае демон *inetd* должен будет запускать новый экземпляр сервера *httpd* для каждого входящего соединения.

Хитрым моментом здесь является указание номера порта. Вы можете захотеть запустить *httpd* без привилегий суперпользователя (то есть у вас может и не быть привилегий *root* на нужной машине, а вы хотите запустить *httpd* от своего имени). В этом случае необходимо использовать порт с номером 1024 или выше. Например, если указать

```
Port 2112
```

то вы можете запустить *httpd* как обычный пользователь. В этом случае HTTP URL к этой машине должен указываться как

```
http://www.ecoveggie.org:2112/…
```

Если номер порта в URL не указан (обычно так и делается), по умолчанию используется порт с номером 80.

Директива

```
DocumentRoot /usr/local/httpd/htdocs
```

указывает каталог, где хранятся документы, предоставляемые по протоколу HTTP. Эти документы написаны на языке HTML.

Например, если кто-либо обратится к URL:

```
http://www.ecoveggie.org/fruits.html
```

то реальным файлом, к которому будет осуществляться доступ, окажется */usr/local/httpd/htdocs/fruits.html*.

Директива `UserDir` определяет каталог, который может быть создан каждым пользователем в своем домашнем каталоге для хранения публично доступных HTML-файлов. Например, если бы мы использовали URL:

```
http://www.ecoveggie.org/~mdw/linux-info.html
```

реально мы обратились бы к файлу *~mdw/public_html/linux-info.html*.

В следующих строках включаются функции указателя *httpd*:

```
# Если URL указан без имени файла, считать, что запрашивается файл
# index (если таковой существует).
DirectoryIndex index.html

# Включить «забавный» индекс каталогов
FancyIndexing on
```

В данном примере, если броузер попытается обратиться к серверу, указав в URL только каталог (без имени файла), ему будет возвращен файл *index.html* из данного каталога, если таковой существует. В противном случае *httpd* генерирует причудливый индекс со значками, представляющими разные типы файлов. Пример такого индекса показан на рис. 5.2.

Значки назначаются директивой `AddIcon`, как показано ниже:

```
# Устанавливает разные значки для необычных индексов по имени файлов
# Например, DocumentRoot/icons/movie.xbm используется для файлов,
# заканчивающихся на .mpg и .qt
AddIcon /icons/movie.xbm .mpg
AddIcon /icons/back.xbm ..
AddIcon /icons/menu.xbm ^^DIRECTORY^^
AddIcon /icons/blank.xbm ^^BLANKICON^^
DefaultIcon /icons/unknown.xbm
```

Имена файлов значков (такие как */icons/movie.xbm*) по умолчанию указываются относительно `DocumentRoot`. (Существуют и другие способы указать путь к документам и значкам, например при помощи псевдонимов. Это обсуждается позже.) Имеются также две директивы, позволяющие указать значок для документа: `AddIconByType` – на основе его типа MIME и `AddIconByEncoding` – на основе его кодировки (например, в зависимости от того, сжат ли документ).

Кроме того, можно указать значок, который используется, если ничто из вышеперечисленного не подходит. Это делается директивой `DefaultIcon`.

Необязательные директивы `ReadmeName` и `HeaderName` указывают имена файлов, включаемых в индекс, генерируемый *httpd*:

```
ReadmeName README
HeaderName HEADER
```

Здесь, если в текущем каталоге есть файл *README.html*, он будет добавлен в индекс. Файл *README* будет добавлен, если *README.html* не существует. Аналогично *HEADER.html* или *HEADER* добавляются в начало индекса, генерируемого *httpd*. Вы можете использовать их для описания содержимого конкретного каталога, когда индекс запрашивается броузером.

```
# Имя локального файла доступа.
AccessFileName .htaccess

# Тип MIME по умолчанию.
DefaultType text/plain
```

Директива `AccessFileName` указывает имя *локального файла доступа* для каждого каталога. (Об этих файлах будет рассказано ниже.) Директива `DefaultType` указывает тип MIME для документов, не перечисленных в *mime.types*.

Следующие строки задают каталоги для полезных файлов:

```
# Устанавливает место расположения значков.
Alias /icons/ /usr/local/html/icons/

# Устанавливает место расположения CGI-программ.
ScriptAlias /cgi-bin/ /usr/local/httpd/cgi-bin/
```

Директива `Alias` указывает псевдоним для путей к любым документам, к которым обращаются по URL или которые перечислены в *srm.conf*. Ранее мы применяли директиву `AddIcon`, чтобы установить имена значков, используя при этом такие имена, как */icons/movie.xbm*. Здесь мы указываем, что имя */icons/* нужно преобразовать в */usr/local/html/icons/*. Там и должны храниться все файлы значков. Вы можете использовать `Alias` для создания и других псевдонимов.

Директива `ScriptAlias` похожа на `Alias`, но устанавливает реальное расположение CGI-сценариев в системе. В данном примере предполагается хранить сценарии в каталоге */usr/local/httpd/cgi-bin/*. Каждый раз, когда в URL первым каталогом указан */cgi-bin/*, он преобразуется в настоящее имя каталога. Подробная информация по CGI и сценариям есть в книге Скотта Гулича (Scott Guelich), Шишира Гундаварама (Shishir Gundavaram) и Гюнтера Бирзнекса (Gunther Birznieks) «CGI Programming with Perl» (O'Reilly)[1].

Элементы `<Directory>` определяют параметры и атрибуты конкретного каталога:

```
# Установить параметры для каталога сценариев cgi-bin.
<Directory /usr/local/html/cgi-bin>
Options Indexes FollowSymLinks
</Directory>
```

Здесь мы указали, что каталог сценариев CGI должен иметь параметры доступа `Indexes` и `FollowSymLinks`. В число возможных параметров доступа входят:

`FollowSymLinks`

Необходимо следовать по символьным ссылкам в данном каталоге для доступа к документам, на которые они указывают. Этот параметр вообще не безопасен при использовании в многопользовательских системах, поскольку он позволяет создавать ссылки на другие файлы и каталоги (например, на файл */etc/passwd*). Параметр `SymLinksIfOwnerMatch` представляет собой менее опасную (хотя чуть более медленную) альтернативу.

`ExecCGI`

Позволить выполнение сценариев CGI из этого каталога.

`Indexes`

Разрешить создание индексов из этого каталога.

---

[1] Скотт Гулич, Шишир Гундаварам и Гюнтер Бирзнекс «CGI программирование на Perl», 2-е издание. – Пер. с англ. – СПб.: Символ-Плюс, 2001.

`None`

Запретить все параметры для этого каталога.

`All`

Разрешить все параметры для данного каталога.

Существуют и другие параметры, подробное описание которых можно найти в документации по *httpd*.

Затем настраиваются весьма строгие ограничения по умолчанию на доступ ко всей файловой системе:

```
# Параметры по умолчанию
<Directory />

# Отключить все имеющиеся возможности
Options None

# Не позволять изменять параметры настройки
# с помощью локальных файлов.
AllowOverride None

# Полностью запретить доступ к любым файлам.
Order allow,deny
Deny from all

</Directory>
```

Мы начали с того, что полностью запретили доступ к файловой системе. Теперь следует явно разрешить доступ к файлам, которые будут обслуживаться с помощью Apache. Как минимум, нам надо разрешить некоторые параметры и атрибуты для каталога */usr/local/httpd/htdocs*, где находятся документы HTML. Следующие настройки определяют параметры доступа к этому каталогу и к вложенным в него подкаталогам:

```
# Настройки доступа к файлам веб-сервера.
<Directory /usr/local/httpd/htdocs>

# Разрешить автоматическое создание индексного файла
# и следование по ссылкам.
Options Indexes SymLinksIfOwnerMatch

# Позволить локальному файлу доступа, .htaccess, переопределять
# любые атрибуты, указанные здесь.
AllowOverride All

# Отменить ограничения на доступ к документам в данном каталоге.
Order allow,deny
allow from all

</Directory>
```

Мы включили параметры `Indexes` и `SymLinksIfOwnerMatch` для данного каталога. Директива `AllowOverride` позволяет локальному файлу доступа для каждого каталога (в данном случае *.htaccess*) переопределять любые атрибуты, указанные здесь. Файл *.htaccess* имеет тот же формат, что и глобальный файл *access.conf*, но применяется только к каталогу, в котором он находится. Таким образом, можно

указать атрибуты для отдельных каталогов, включив в каждый из них файл *.htaccess*, вместо того, чтобы перечислять их в глобальном файле с настройками.

Локальные файлы доступа в основном применяются для того, чтобы дать возможность пользователям устанавливать разрешения на доступ к личным каталогам HTML (таким как *~/public_html*), не обращаясь к администратору с просьбой изменить глобальный файл с настройками параметров доступа. Однако при этом возникает проблема безопасности. Например, пользователь может позволить в своем каталоге исполнение CGI-сценариев на стороне сервера. Если вы отключите `AllowOverride`, пользователи не смогут обойти параметры доступа, указанные в глобальном файле *access.conf*. Это можно сделать, указав:

```
AllowOverride None
```

что отключит действие локальных файлов с настройками параметров доступа *.htaccess*.

Поле `<Limit GET>` используется с целью определения правил доступа для броузеров, пытающихся получить документы с данного сервера. В нашем случае мы указали `Order allow,deny`. Это означает, что правила, разрешающие (`allow`) доступ, рассматриваются перед запрещающими (`deny`) правилами. Затем мы пишем правило `allow from all`, которое означает, что любой хост может обращаться к документам на этом сервере. Если вы хотите запретить доступ из конкретных доменов или систем, можно добавить строку:

```
deny from .nuts.com biffnet.biffs-house.us
```

Первая запись запрещает доступ всем машинам из домена `nuts.com`. Вторая запись запрещает доступ с сайта `biffnet.biffs-house.us`.

srm.conf и access.conf

Файлы *srm.conf* и *access.conf* следует оставить пустыми. В прежних версиях Apache в файле *srm.conf* (от «Server Resource Map» – карта ресурсов сервера) перечислялись возможности, предоставляемые сервером, а *access.conf* управлял доступом к файлам Apache. Все, что раньше находилось в этих файлах, теперь содержится в файле *httpd.conf*.

Запуск httpd

Теперь мы готовы запустить *httpd* и позволить нашей машине обслуживать запросы HTTP. Как указывалось ранее, можно запустить *httpd* из *inetd* или как автономный сервер. Здесь мы опишем, как запустить *httpd* в автономном режиме.

Все, что необходимо для запуска *httpd*, – это выполнить команду:

```
httpd -f configuration file
```

где *configuration file* – это путь к *httpd.conf*. Например, команда

```
/usr/sbin/httpd -f /etc/httpd/httpd.conf
```

запустит *httpd* с настройками из файла */etc/httpd/httpd.conf*.

Ошибки при запуске сервера или при обращении к документам регистрируются в журналах *httpd* (их местонахождение определено в *httpd.conf*). Помните, что

вы должны запускать *httpd* как *root*, если используется порт с номером 1023 и меньше. Когда вы отладите работу *httpd*, его можно будет запускать автоматически, включив соответствующую строку с командой *httpd* в один из ваших системных *rc*-файлов, например */etc/rc.d/rc.local*.

В некоторые версии Apache входит утилита *apachectl*, с помощью которой гораздо удобнее выполнять запуск, остановку и перезапуск сервера *httpd*. Например, команда

```
apachectl configtest
```

позволяет проверить корректность файла с настройками без запуска сервера. Наконец, следует упомянуть, что запускать, перезапускать и останавливать Apache можно с помощью сценария */etc/init.d/apache*, передавая ему один из параметров: `start`, `restart` или `stop`.

Конечно, для запроса документов по HTTP из броузера их надо еще написать. Эта тема выходит за рамки данной книги. Рекомендуем книги по HTML, изданные O'Reilly: «HTML & XHTML. The Definitive Guide»[1] Чака Муссиано (Chuck Musciano) и Билла Кеннеди (Bill Kennedy) и «HTML Pocket Reference» Дженнифер Нидерст (Jennifer Niederst). О веб-странице особого рода, заполняемой данными из базы, рассказывается в главе 25 данной книги.

[1] Чак Муссиано, Билл Кеннеди «HTML и XHTML», 6-е издание. – Пер. с англ. – СПб: Символ-Плюс, 2008.

23

Транспортировка и обработка сообщений электронной почты

Электронная почта (email) – это одна из самых желанных функций компьютерной системы. Обмениваться сообщениями могут локальные пользователи системы Linux или пользователи с различных хостов сети. Чтобы организовать электронную почту, нужны программы трех типов: *mail user agent*, или *почтовый клиент* (MUA), *mail transport agent* (MTA), или *агент передачи почты*, и *transport protocol*, или *транспортный протокол*.

Почтовый клиент предоставляет интерфейс для чтения почты, создания новых сообщений и хранения сообщений. Linux предлагает большой выбор почтовых клиентов, которые постоянно совершенствуются. Некоторые почтовые клиенты предоставляют особые возможности, например, их можно использовать в качестве клиента новостей или как веб-броузер.

Почтовые клиенты различаются по степени поддержки MIME. (MIME расшифровывается как Multimedia Internet Mail Extensions. Это скорее общий стандарт для описания содержимого электронных сообщений, чем ориентированный на мультимедиа.) Однако давать какие-либо рекомендации сложно, поскольку во всех почтовых клиентах поддержка MIME непрерывно улучшается. Кроме того, проблемы чаще возникают не с программным обеспечением электронной почты, а с необходимостью увязывать типы MIME с правильными приложениями просмотра и обработки.

Почтовый клиент для передачи почты между пользователями (как локальными, так и между системами) полагается на MTA. MTA, в свою очередь, использует для передачи почты транспортный протокол – обычно UUCP (UNIX-to-UNIX Copy) либо SMTP (Simple Mail Transport Protocol – простой протокол передачи почты).

Существует несколько возможных сценариев использования электронной почты в Linux-системах, и в зависимости от этих сценариев требуется установка разных наборов программного обеспечения. Однако почтовый клиент потребуется в любом случае.

Первый сценарий применяется при доступе к Интернету по коммутируемой линии через провайдера (ISP). Обычно в этом случае на Linux-машине есть только один пользователь, хотя это не обязательно. Провайдер принимает вашу почту

из Интернета и хранит ее на своих жестких дисках. Вы можете забрать свою почту в любое время, используя обычный протокол POP3 (Post Office Protocol) или более новый протокол IMAP (Internet Message Access Protocol). Исходящая почта в этом случае практически всегда посылается по протоколу SMTP, который повсеместно используется для передачи почты в Интернете.

В простейшем случае почтовый клиент используется как для получения почты по POP3 или IMAP, так и для отправки ее по SMTP. При этом даже не надо устанавливать MTA, так как почтовый клиент возьмет на себя всю работу. Этот вариант не отличается большой гибкостью, но он вполне может подойти, если нужно просто получить доступ к электронной почте. Эту схему работы поддерживают такие почтовые клиенты, как KMail из KDE и почтовая программа, встроенная в Mozilla (описываются далее).

Почтовые клиенты, построенные на основе броузера, такие как *gmail* или GMX, – это совсем другая история. Они предназначены для взаимодействия с почтовым ящиком, который хранится на сервере, доступ к этому ящику может осуществляться по протоколам POP3 или IMAP, нередко автоматически. Современные почтовые клиенты, построенные на основе броузера, обычно используют протокол IMAP.

Если нужна большая гибкость (за нее, конечно, придется заплатить трудом по настройке и управлению), можно установить MTA, например Postfix, который описан в следующем разделе. Необходимо установить программу, которая будет забирать почту с сервера POP3 или IMAP, находящегося у провайдера. Эта программа будет забирать почту по вашему требованию и передавать сообщения работающему на вашей системе MTA, который распределит почту по почтовым папкам получателей. Именно это делает программа *fetchmail*, которую мы рассмотрим в этой главе. Исходящая почта по-прежнему посылается по протоколу SMTP, но MTA на вашей системе позволяет отсылать почту не сразу на SMTP-сервер провайдера, а на собственный сервер, предоставляемый MTA. Затем MTA пересылает почту провайдеру. Такая схема работы позволяет настроить MTA на периодическую отсылку почты, чтобы не подключаться каждый раз к интернет-провайдеру.

Третий сценарий предназначен для машин с постоянным подключением к Интернету, либо через шлюз по сети Ethernet, либо по выделенной линии. В этом случае вы, возможно, захотите получать почту, как только она появится у провайдера, не сохраняя ее там. Это также требует настройки MTA. Входящая почта будет направлена вашему SMTP-серверу (то есть MTA). Ваш провайдер должен также настроить свою систему соответствующим образом.

Существует, конечно, еще много сценариев для работы с почтой, и возможны также комбинации из трех вышеперечисленных. Если вы собираетесь настраивать почтовый сервис для целой сети, необходимо прочитать «Linux Network Administrator's Guide»[1], а также книгу о вашем MTA.

У вас есть большой выбор между программами для настройки электронной почты на Linux. Мы не можем рассказать обо всех доступных решениях, но опишем некоторые наиболее часто используемые пакеты, неплохо подходящие для сво-

[1] Олаф Кирк «Linux для профессионалов. Руководство администратора сети». – Пер. с англ. – СПб.: Питер, 2000.

их задач. Почтовые программы для конечных пользователей, такие как Kmail и Evolution, уже были описаны в предыдущих главах. В этой главе будет рассказываться о таких популярных решениях для Linux, существующих в настоящее время, как агент передачи почты Postfix и реализация протоколов POP3 и IMAP в программе *fetchmail*. Эти пакеты относительно просты в настройке, но предоставляют все возможности, необходимые большинству пользователей. Эти средства охватывают все три описанные выше сценария.

Postfix MTA – агент передачи почты

Есть несколько MTA, работающих под Linux. Исторически самым распространенным пакетом под UNIX был *sendmail*, появившийся на сцене довольно давно. Обычно его считают более сложным в использовании, чем остальные, но он тщательно документирован в книге «Sendmail» Брайана Косталеса (Bryan Costales) и Эрика Олмена (Eric Allman), O'Reilly.

Postfix – более новый MTA[1], разработанный гуру в области безопасности Витсе Венема (Wietse Venema) в качестве замены для *sendmail*. Он спроектирован как совместимый с *sendmail*, но обеспечивающий более высокий уровень защиты и более простой в настройке.

Postfix – очень гибкий и надежный программный продукт, имеющий несколько уровней защиты от потенциальных злоумышленников. Postfix также создавался с расчетом на достижение высокой производительности и использует технологии, ограничивающие такие замедляющие работу операции, как создание новых процессов и обращение к файловой системе. Его проще настраивать и администрировать, чем другие почтовые пакеты, поскольку в нем используются простые файлы конфигурации и справочные таблицы для перезаписи адресов. Он отличается простотой в использовании в качестве базового MTA при сохранении возможности применения в более сложной среде.

Postfix встраивается во многие дистрибутивы Linux, поэтому возможно, что он уже установлен в вашей системе. Если это не так, можно найти готовый пакет или скомпилировать его самостоятельно из исходных текстов. На домашней странице Postfix (*http://www.postfix.org/*) есть ссылки для загрузки исходных текстов (*Download*) и пакетов для различных дистрибутивов Linux (*Packages and Ports*).

Postfix выпускается в двух сериях версий: официальной и экспериментальной. Экспериментальные версии содержат все свежие патчи и новые функциональные возможности, которые могут измениться перед включением в официальный релиз. Термин «экспериментальный» не должен вас отпугивать: эти версии очень стабильны и тщательно протестированы. Если вам нужна функция, имеющаяся только в экспериментальной версии, вы вполне можете воспользоваться ею. Прочтите комментарии к версиям в обеих сериях, которые объяснят, в чем различия между ними.

---

[1] Ральф Гильдебрандт и Патрик Кеттер «Postfix. Подробное руководство». – Пер. с англ. – СПб.: Символ-Плюс, 2008.

Некоторые сведения о DNS

Прежде чем устанавливать Postfix, нужно понять, что, если ваша машина будет получать почту с других машин через Интернет, необходимо правильно настроить DNS для вашего домена. О DNS рассказывается в главе 13.

Предположим для целей нашего рассказа, что вы настраиваете хост с именем `halo` в домене *example.org*, и что на вашей машине есть учетная запись пользователя *michael*. Независимо от того, каким способом вы собираетесь получать почту, у вашего хоста *halo.example.org* должна быть A-запись DNS, отображающая это имя хоста в его IP-адрес.

В данном примере вашим почтовым адресом будет *michael@halo.example.org* или *michael@example.org*. Если вы хотите использовать адрес первого вида, то для получения сообщений достаточно настроить A-запись DNS.

Если ваша машина должна получать всю почту для *example.org* (*michael@example.org*), домен должен обладать MX-записью DNS, указывающей на ваш хост *halo.example.org*. Если вы самостоятельно настраиваете DNS для своего домена, обязательно прочтите документацию, чтобы понять, как это работает. Либо обсудите со своим администратором DNS или интернет-провайдером маршрутизацию почты на вашу машину.

Postfix часто использует DNS в обычной работе и пользуется соответствующими библиотеками Linux для выполнения запросов DNS. Вы должны правильно настроить свою систему для выполнения поиска в DNS (см. раздел «Настройка DNS» в главе 13). Postfix обычно должен найти MX-запись, чтобы осуществить доставку. Не следует рассчитывать, что когда Postfix сообщит о проблеме DNS с адресом, а вы обнаружите, что распознавание домена происходит правильно, то доставка почты пройдет успешно. Если Postfix сообщает о проблеме, можно быть почти уверенным, что проблема действительно есть.

Установка Postfix

Несмотря на наличие готовых дистрибутивов, может потребоваться самостоятельно скомпилировать пакет, чтобы использовать дополнительные библиотеки или функции, отсутствующие в вашем дистрибутиве. Возможно также, что вам потребуется последняя версия, чтобы получить доступ к новой функции, еще не включенной в ваш дистрибутив.

Прежде чем устанавливать Postfix, учтите, что в этот пакет входят три команды: */usr/bin/newaliases*, */usr/bin/mailq* и */usr/sbin/sendmail*, которые обычно используются *sendmail*. Postfix заменяет эти команды другими, которые работают с системой Postfix, а не с *sendmail*. Следует переименовать существующие команды *sendmail*, чтобы они не были уничтожены при установке Postfix и вы могли при желании снова воспользоваться первоначальными исполняемыми модулями Sendmail:

```
# mv /usr/bin/newaliases /usr/bin/newaliases.orig
# mv /usr/bin/mailq /usr/bin/mailq.orig
# mv /usr/sbin/sendmail /usr/sbin/sendmail.orig
```

Postfix использует файлы базы данных UNIX для хранения своей информации о псевдонимах и справочной таблице, поэтому перед компиляцией Postfix необ-

ходимо установить библиотеки *db*. Эти библиотеки есть в RPM-пакете *db-devel* или пакете Debian *libdb4.3-dev*. Если вы не пользуетесь менеджером пакетов, то можете получить библиотеки непосредственно у Sleepycat Software (*http://www.sleepycat.com/*). Если вы пользуетесь RPM, выполните следующую команду и проверьте, установлены ли в вашей системе необходимые библиотеки:

```
# rpm -qa | grep db-devel
db3-devel-4.3.27-3
```

Вы должны увидеть нечто похожее на вторую строку, в которой показан пакет *db-devel* и номер его версии. Если *rpm* ничего не возвратит, нужно сначала установить эти библиотеки и только потом устанавливать Postfix.

В Debian узнать, установлены ли библиотеки, можно с помощью *dpkg*:

```
# dpkg -l libdb4.3-dev
```

Если вы загрузили двоичный дистрибутив Postfix, воспользуйтесь своим менеджером пакетов (описанным в главе 12) для его установки. Если вы загрузили файл с исходными текстами *postfix-2.2.5.tar.gz*, переместите его в подходящий каталог (например, в свой домашний) и распакуйте. Цифры в имени файла указывают версию пакета. У вас могут оказаться другие цифры в зависимости от того, какую версию вы загрузите.

Для компиляции Postfix следуйте описанной ниже процедуре. Учтите, что вам потребуются права *root*, чтобы создать пользователя и группу и установить пакет.

1. Переименуйте свои исполняемые файлы *sendmail*, как описано ранее.
2. Создайте пользователя с именем *postfix* и группу с именем *postdrop*. Процедура создания учетных записей пользователей и групп описана в разделе «Управление учетными записями пользователей» главы 11.
3. Выполните *gunzip* над сжатым файлом, получив файл с именем *postfix-2.2.5.tar*.
4. Выполните:

   ```
   tar -xvf postfix-2.2.5.tar
   ```

   чтобы распаковать исходные тексты код в каталог *postfix-2.2.5*.
5. Перейдите в каталог, созданный при распаковке файла. Вы обнаружите в нем файл *INSTALL*, содержащий подробные инструкции по компиляции системы Postfix. Обычно для этого требуется лишь ввести команду `make`, находясь в этом каталоге.
6. Если компиляция выполнится без ошибок, введите *make install*, чтобы установить Postfix на вашей машине. В ответ на приглашения сценария установки следует принимать все значения по умолчанию.

После установки файлы Postfix будут размещены в следующих каталогах:

/usr/libexec/postfix

В этом каталоге содержатся различные демоны Postfix. Postfix использует расщепленную архитектуру, в которой несколько отдельных программ выполняют разные задачи. Первым запускается демон *master*. Он занимается запуском других программ по мере необходимости. По большей части вас не

должно беспокоить, какие программы здесь находятся. Запуск и останов Postfix осуществляется командой *postfix*, расположенной в каталоге */usr/sbin*.

/etc/postfix

Обычно в этом каталоге находятся десятки файлов конфигурации Postfix, но Postfix использует только *master.cf* и *main.cf*, а также несколько справочных таблиц. Остальные файлы служат примерами, документирующими различные параметры, используемые для настройки.

Файл *master.cf* управляет различными процессами Postfix. В нем есть строка для каждого компонента Postfix. Структура файла описана в комментариях, находящихся в самом файле. Обычно для простой установки Postfix не требуется делать никаких изменений.

Файл *main.cf* является глобальным файлом с настройками SMTP. Он содержит список параметров, для которых установлено одно или несколько значений в формате:

```
parameter = value
```

Комментарии обозначаются символом решетки (#) в начале строки. Комментарии нельзя помещать на одной строке с параметрами. Строки комментариев могут начинаться с пробелов или табуляции, но должны быть на отдельных строках.

Если для параметра устанавливаются несколько значений, они разделяются запятыми или пробельными символами (включая перевод строки), но если параметр описывается в нескольких строках, вторая и последующие строки должны начинаться с пробельного символа. Значения могут ссылаться на другие параметры, при этом имя параметра предваряется знаком доллара ($).

Ниже приводится пример записи, содержащей комментарий, несколько строк и ссылку на параметр:

```
# Список всех машин, с которых я жду почту.
mynetworks = $myhostname
    192.168.75.0/24
    10.110.12.15
```

/usr/sbin

Все команды Postfix находятся в */usr/sbin*, и их имена начинаются с `post`. Существуют команды для создания файлов индексов, управления почтовой очередью и других задач администрирования системы Postfix. Здесь находится команда *postfix*, используемая для останова и запуска Postfix (описывается ниже).

/var/spool/postfix

Менеджер очередей Postfix – важный компонент системы Postfix, который принимает входящие почтовые сообщения и взаимодействует с другими компонентами Postfix для их доставки. Он хранит свои файлы в каталоге */var/spool/postfix*. Очереди, которыми он управляет, перечислены ниже. Postfix предоставляет несколько утилит для управления очередями, например *postcat*, *postsuper* и *mailq*, но для просмотра очереди можно воспользоваться обычными командами Linux, такими как *find* и *cat*.

/var/spool/postfix/incoming

Все входящие сообщения, полученные через сеть или локально.

/var/spool/postfix/active

Сообщения, которые менеджер очередей доставляет в данное время или готовится доставить.

/var/spool/postfix/deferred

Сообщения, которые не удалось доставить немедленно. Postfix будет пытаться доставить их снова.

/var/spool/postfix/corrupt

Сообщения, которые совершенно невозможно прочесть, либо они повреждены как-то иначе, и доставить их невозможно. Они хранятся здесь, чтобы при необходимости можно было посмотреть на них и выяснить, в чем проблема. Эта очередь используется редко.

/usr/local/man

Postfix устанавливает документацию в виде страниц руководства. В документации описываются утилиты командной строки, демоны и файлы с настройками.

Как уже отмечалось, Postfix устанавливает новые файлы в каталоги */usr/bin/newaliases*, */usr/bin/mailq* и */usr/sbin/sendmail*.

Настройка Postfix

Перед тем как впервые запустить Postfix, нужно проверить правильность формата таблицы псевдонимов и определения некоторых критически важных параметров в настройках.

В *sendmail* исторически использовался файл */etc/aliases*, который описывал отображение одних локальных имен пользователей в другие. Postfix продолжает эту традицию. Файл */etc/aliases* – это обычный текстовый файл, используемый в качестве входных данных при создании файла индексированной базы данных для быстрого поиска псевдонимов в системе. Есть, по крайней мере, два важных псевдонима, которые должны быть установлены в файле */etc/aliases*. Если раньше у вас на машине работала *sendmail*, то эти псевдонимы, возможно, уже правильно установлены, но убедитесь, что в вашем файле есть записи для *root* и *postmaster*, указывающие на реального пользователя, получающего почту в вашей системе. После проверки псевдонимов выполните команду *newaliases*, которая перестроит файл индекса в формате, необходимом для Postfix.

Файл */etc/postfix/main.cf* содержит много параметров, но среди них есть лишь несколько важных, которые нужно проверить, прежде чем запускать Postfix. Мы расскажем о них в этом разделе. Если вы установили Postfix из готового дистрибутива, то эти параметры, возможно, уже правильно установлены. Не исключено, что значения Postfix по умолчанию окажутся действенными в вашей системе, но для верности отредактируйте файл */etc/postfix/main.cf*.

myhostname

Это полное имя хоста для вашей системы. По умолчанию Postfix использует имя, которое возвращает функция `gethostname`. Если это значение не полностью квалифицировано и вы не установили этот параметр, Postfix не запустится. Проверить это имя можно, выполнив команду *hostname*. Будет правильным задать здесь явно полное имя своего хоста:

```
myhostname = halo.example.org
```

mydomain

Задает имя домена для этой системы. Это значение используется затем по умолчанию в других местах. Если не задать его явно, Postfix использует доменную часть `myhostname`. Если вы установили `myhostname`, как показано выше, и *example.org* соответствует вашей системе, задавать этот параметр не обязательно.

mydestination

Задает список доменных имен, для которых данная система должна принимать почту. Иными словами, значением этого параметра должна быть доменная часть почтовых адресов, для которых вы хотите получать почту. По умолчанию Postfix использует значение, указанное в `myhostname`. Если вы настраиваете свою систему на прием почты для всего своего домена, задайте само имя домена. В качестве значения этого параметра можно использовать переменные `$myhostname` и `$mydomain`:

```
mydestination = $myhostname $mydomain
```

myorigin

Этот параметр используется для добавления имени домена в сообщения, отправляемые локально, если оно в них отсутствует. Например, если пользователь вашей системы отправляет сообщение, содержащее в поле `From:` только локальное имя пользователя, Postfix дописывает это значение к локальному имени. По умолчанию Postfix использует `myhostname`, но если ваша машина обрабатывает почту всего домена, вместо этого нужно задать `$mydomain`:

```
myorigin = $mydomain
```

Некоторые дистрибутивы Linux, содержащие Postfix, настраивают его на использование по умолчанию Procmail. Программа Procmail представляет собой отдельный агент доставки почты (mail delivery agent, MDA), который может фильтровать и сортировать почту при доставке индивидуальным пользователям вашей системы. Программа Procmail будет описана далее в этой главе. Если вам нужны возможности, предоставляемые Procmail, следует тщательно изучить документацию по этой программе, чтобы разобраться, как она взаимодействует с Postfix. Во многих системах, которые не осуществляют фильтрацию почты на уровне MTA, Procmail образует ненужный дополнительный уровень сложности, поскольку Postfix тоже может осуществлять локальную доставку и обеспечивает часть таких же функций. В вашем дистрибутиве использование Procmail может быть задано в параметрах `mailbox_command` или `mailbox_transport`. Если вы хотите, чтобы Postfix непосредственно выполнял локальную доставку, можете смело закомментировать любой из этих параметров в своем файле */etc/postfix/main.cf*.

Запуск Postfix

После проверки важных параметров конфигурации, описанных выше, и перестройки индексного файла псевдонимов можно запускать Postfix. Зарегистрировавшись как суперпользователь, выполните команду:

```
postfix start
```

Остановить Postfix можно командой:

```
postfix stop
```

После модификации каких-либо файлов с настройками Postfix необходимо перезагрузить выполняющийся образ Postfix, выполнив команду:

```
postfix reload
```

Когда работает Postfix, все пользователи на вашей системе должны быть в состоянии отправлять и принимать сообщения электронной почты.

Если у вас есть приложения, зависящие от *sendmail*, они должны сохранить работоспособность, и пользоваться командой *sendmail* можно, как обычно. Можно направлять ей сообщения из сценариев через канал и выполнять *sendmail -q* для очистки очереди. Эквивалентной командой Postfix для очистки очереди служит *postfix flush*. Параметры *sendmail*, относящиеся к выполнению ее в качестве демона и установке задержки очереди, не работают, поскольку эти функции не поддерживаются командой *sendmail* в Postfix. Все параметры Postfix устанавливаются в двух файлах конфигурации. Многие параметры относятся к очередям Postfix. Их описание можно найти на странице справочного руководства для *qmgr*(8).

Ведение журналов в Postfix

После запуска или перезагрузки Postfix нужно проверить по журналу, не сообщает ли Postfix о каких-либо проблемах. (В большинстве дистрибутивов Linux для этих целей используется файл */var/log/maillog*, но можно проверить это по файлу */etc/syslog.conf*.) Последние сообщения Postfix можно вывести командой *tail /var/log/maillog*. Поскольку процесс Postfix выполняется длительное время, неплохо периодически просматривать журнал, даже если вы не перезагружали программу. Узнать, не было ли во время работы Postfix каких-нибудь важных сообщений, можно с помощью команды:

```
egrep '(reject|warning|error|fatal|panic):' /var/log/maillog
```

Обычно Postfix информирует о том, что происходит в системе, передавая массу полезных сведений демону *syslogd*. В Linux *syslogd* по умолчанию осуществляет запись синхронно, то есть после каждой записи в файл журнала выполняется синхронизация, переносящая все содержимое памяти на диск. В результате может снижаться производительность Postfix (и других процессов). Изменить этот режим по умолчанию можно, если указать дефис перед именем файла журнала в */etc/syslog.conf*. Запись в *syslog.conf* для ведения журнала работы с почтой должна выглядеть так:

```
mail.*          -/var/log/maillog
```

После внесения изменений в файл с настройками необходимо, чтобы *syslogd* заново прочел его. Повторную инициализацию можно осуществить командой *killall -HUP syslogd*.

Запуск Postfix во время загрузки системы

Благодаря совместимости Postfix с *sendmail*, если система настроена для запуска *sendmail* во время инициализации, то, скорее всего, Postfix правильно запустится при начальной загрузке системы. Однако останов системы может не произойти корректно. Большинство дистрибутивов Linux останавливают *sendmail*, находя процесс с именем *sendmail* и останавливая его. Однако процессы Postfix, будучи во многом совместимыми с *sendmail*, не выполняются под именем *sendmail*, поэтому такой останов не срабатывает.

Если вы хотите, чтобы система завершала работу корректно, следует создать свой *rc*-сценарий для Postfix, как описано в разделе «Файлы rc» главы 17. Для запуска и останова Postfix нужно включить в этот сценарий те же самые команды, которые вы выполняете в командной строке, то есть *postfix start* и *postfix stop*. Ниже приведен пример сценария, который можно принять за основу. Может потребоваться посмотреть другие *rc*-сценарии в вашей системе, чтобы выяснить, не нужны ли дополнительные системные проверки или выполнение других соглашений, а потом внести в этот пример необходимые изменения:

```
#!/bin/sh
PATH=""
RETVAL=0

if [ ! -f /usr/sbin/postfix ] ; then
    echo "Unable to locate Postfix"
    exit 1
fi
if [ ! -f /etc/postfix/main.cf ] ; then
    echo "Unable to locate Postfix configuration"
    exit 1
fi

case "$1" in
    start)
        echo -n "Starting Postfix: "
        /usr/sbin/postfix start > /dev/null 2>1
        RETVAL=$?
        echo
        ;;
    stop)
        echo -n "Stopping Postfix: "
        /usr/sbin/postfix stop > /dev/null 2>1
        RETVAL=$?
        echo
        ;;
    restart)
        echo -n "Restarting Postfix: "
        /usr/bin/postfix reload > /dev/null 2>1
        RETVAL=$?
        echo
```

```
        ;;
    *)
        echo "Usage: $0 {start|stop|restart}"
        RETVAL=1
esac
exit $RETVAL
```

Поместите этот сценарий в */etc/rc.d/init.d* или */etc/init.d*, в зависимости от того, каким дистрибутивом Linux вы пользуетесь. Затем создайте необходимые символические ссылки в каждом из каталогов *rcN.d* для каждого уровня исполнения, на котором нужно запускать Postfix (см. раздел «Файлы init, inittab и rc» в главе 17). Например, если вы хотите запускать Postfix на уровнях исполнения 3 и 5 и останавливать на уровнях исполнения 0 и 6, создайте символические ссылки, как указано ниже, для Red Hat. В Debian каталоги *rcN.d* находятся непосредственно в */etc*:

```
# cd /etc/rc.d/rc3.d
# ln -s ../init.d/postfix S97postfix
# cd /etc/rc.d/rc5.d
# ln -s ../init.d/postfix S97postfix
# cd /etc/rc.d/rc0.d
# ln -s ../init.d/postfix K97postfix
# cd /etc/rc.d/rc6.d
# ln -s ../init.d/postfix K97postfix
```

Если вы создаете *rc*-сценарий для Postfix, измените настройки системы, чтобы *sendmail* не запускалась при инициализации.

Управление ретрансляцией в Postfix

Установка по умолчанию позволяет любой системе в вашей подсети ретранслировать почту через ваш почтовый сервер. Если вы хотите изменить эту установку по умолчанию, то можете указать в параметре `mynetworks` список хостов или сетей, которым вы разрешаете ретранслировать почту через вашу машину. Вы можете задать список IP-адресов или шаблоны сети (либо маски сети), чтобы соответствующие им клиенты SMTP могли ретранслировать почту. Перечисленные сети или IP-адреса могут быть любыми. Например, если вы хотите ретранслировать почту со своей машины на работе через систему Postfix, установленную у вас дома, можете задать IP-адрес своей машины на работе в конфигурации Postfix дома.

Ниже приводится пример разрешения ретрансляции почты из локальной подсети (192.168.75.0/28) и еще одного хоста, расположенного где-то в другом месте:

```
mynetworks = 192.168.75.0/28 10.150.134.15
```

Если вы хотите разрешить ретрансляцию для мобильных пользователей, у которых нет статических IP-адресов, нужно воспользоваться каким-то из механизмов аутентификации SMTP. Postfix может работать с аутентификацией SASL (для чего требуется компиляция Postfix с дополнительными библиотеками и специальная настройка программного обеспечения клиентов) и с pop-before-smtp (что требует работы сервера POP на той же машине для первоначальной аутентификации пользователей).

Важно, чтобы ретрансляция была разрешена только тем пользователям, которым вы доверяете. На заре развития Интернета открытая ретрансляция встречалась сплошь и рядом. К сожалению, широкое распространение спама послужило причиной ликвидации таких свобод. Если ваш MTA не защищен, вы подвергаете себя и другие системы в Интернете опасности злоупотреблений. Распространители спама постоянно разыскивают открытые ретрансляторы, и если вы допустите, что он появится в сети, то вскоре его обнаружат. К счастью, установка Postfix по умолчанию настроена на правильное поведение. Однако когда в настройки Postfix вносится много изменений (особенно, как это ни забавно, при настройке для борьбы со спамом), можно случайно открыть систему для ретрансляции. В Интернете существуют некоторые инициативы борьбы со спамом в режиме онлайн, которые предлагают проверить правильность настроек сервера, направленных против бесконтрольной ретрансляции почты. Попробуйте, например, *http://www.abuse.net/relay.html*.

Если вы хотите, чтобы ваша собственная система Postfix могла ретранслировать почту через другой MTA, задайте IP-адрес сервера ретрансляции с помощью параметра `relayhost`. Postfix обычно самостоятельно определяет, куда доставлять сообщения, основываясь на адресе получателя. Однако если, например, ваша система защищена брандмауэром, может потребоваться, чтобы Postfix передавал все сообщения другому почтовому серверу, осуществляющему фактическую доставку. Когда вы указываете сервер-ретранслятор, Postfix обычно выполняет запрос DNS для получения адреса почтового ретранслятора (MX) этой системы. Можно отменить этот поиск в DNS, заключив имя хоста в квадратные скобки:

```
relayhost = [mail.example.org]
```

Дополнительные настройки

Описанная выше настройка создает простую установку Postfix для отправки и получения почты пользователями вашей системы. Но Postfix является чрезвычайно гибким MTA с большим количеством параметров конфигурации, в число которых входит поддержка множественных виртуальных доменов, почтовых списков рассылки, блокировка спама и поиск вирусов. Информацию о более сложных видах настройки можно найти на страницах справочного руководства, в файлах HTML и примерах файлов конфигурации, поставляемых с Postfix.

Procmail

Личностям, известным в Интернете, уделяется ничуть не меньшее внимание, чем знаменитостям в реальной жизни. Любой желающий может стать известным человеком в Интернете, для этого достаточно лишь подписаться на какой-нибудь публичный список рассылки или открыть свою домашнюю страничку, и дело в шляпе. Однако после этого он будет находиться под пристальным вниманием спамеров, рассылающих огромное количество писем с предложениями разбогатеть, увеличить определенные части тела или приобрести материальные ценности, если пожелает помочь получить их из Ирака.

Спасти свою почту от спама вам поможет пара гвардейцев по имени Procmail и SpamAssassin. Procmail – это фильтр почты общего назначения, в то время как

SpamAssassin – специализированный фильтр, предназначенный для борьбы со спамом, червями, вирусами и тому подобным мусором. В этом разделе будет рассматриваться Procmail, а в следующем – SpamAssassin.

Основные понятия Procmail

Для понимания основных принципов Procmail сначала необходимо разобраться с тем, когда он вызывается. Обычно входящая почта обрабатывается в такой последовательности: когда поступает новое почтовое общение, клиент электронной почты вызывает Procmail, передавая ему в качестве аргумента это почтовое отправление. В большинстве программ фильтрации под термином *«фильтр»* или *«правило»* подразумевается набор условий, на соответствие которым проверяется почта, и действий, которые должны быть применены к сообщениям, если они этим условиям соответствуют (например, переместить их в определенную папку). В Procmail эти наборы называются *рецептами* (*recipes*), данный термин мы и будем использовать на протяжении всего раздела при описании наборов взаимосвязанных условий и действий. Procmail следует по всем рецептам, один за другим, пока почтовое отправление не будет помечено как доставленное. Если ни один из рецептов не блокирует почту, она доставляется в папку входящих сообщений, как если бы Procmail вообще никак не участвовал в процессе.

Каждый рецепт состоит из двух частей: набора условий и набора действий. Действия, предусматриваемые рецептом, выполняются только в случае, если оказались соблюдены все условия. Кроме того, в результате следования рецепту почтовое отправление может оказаться помеченным как доставленное, о чем уже говорилось выше.

Примерами условий могут служить следующие обстоятельства:

- Письмо пришло с адреса president@whitehouse.gov.
- Тема сообщения: KimDaBa.
- В теле сообщения имеется строка The KDE Image Database.
- Все, перечисленные выше.

Примеры действий:

- Ответить отправителю, что вы находитесь в отпуске.
- Переслать письмо другому получателю.
- Сохранить письмо в файле.
- Изменить некоторые части письма (например, добавить новое поле в заголовок, вставить какой-либо текст и т. п.).

Прежде чем погрузиться в детальное обсуждение темы этого раздела, необходимо спросить себя, действительно ли Procmail так необходим вам. Многие почтовые клиенты, например KMail, позволяют сортировать почту, а кроме того, они намного проще в использовании, чем Procmail. Ниже приводится список обстоятельств, когда использование Procmail действительно можно считать необходимым:

- Когда в использовании находятся самые разнотипные почтовые клиенты. Например, находясь в дороге, вы можете пользоваться веб-интерфейсом для

доступа к почтовому ящику, а дома работать с обычным почтовым клиентом, таким как KMail или mutt.

- Когда отфильтровать почту необходимо еще до того, как она будет загружена почтовым клиентом. Например, чтобы сгенерировать ответ автоматически, который сообщит, что вас нет в офисе.
- Когда объемы входящей почты настолько велики, что ее желательно было бы отфильтровать еще до того, как почтовый клиент загрузит ее (клиент может выполнять фильтрацию чрезвычайно медленно).

Подготовка Procmail к использованию

Procmail входит в состав большинства современных дистрибутивов Linux, но конкретно у вас его может не оказаться, в этом случае посетите сайт *http://www.procmail.org*, где кроме всего прочего можно найти обширную коллекцию всевозможных рецептов.

После того как Procmail будет установлен в системе, можно проверить, вызывается ли он почтовым клиентом. Самый простой способ выполнить такую проверку – это поместить следующий файл *.procmailrc* в домашний каталог и отправить электронное письмо самому себе.

```
SHELL=/bin/sh
MAILDIR=${HOME}/Mail
LOGFILE=${MAILDIR}/procmail.log
LOG="--- Logging ${LOGFILE} for ${LOGNAME}, "
```

Если каталог *~/Mail* отсутствует, его необходимо создать, чтобы данный сценарий смог работать. Если входящая почта хранится в каком-то другом месте, нужно заменить строку `${HOME}/Mail` соответствующим именем каталога. Кроме того, обязательно нужно проверить наличие файла */bin/sh* (скорее всего, он существует), в противном случае нужно внести исправления в сценарий.

Если по умолчанию Procmail вызывается, только что продемонстрированный сценарий должен оставить в файле *~/Mail/procmail.log* запись примерно следующего содержания:

```
--- Logging /home/test/Mail/procmail.log for test, From blackie@blackie.dk Fri
Mar 18 12:25:23 2005
Subject: Fri Mar 18 12:25:22 CET 2005
Folder: /var/spool/mail/test
```

Если этот файл так и не появился после получения письма, отправленного самому себе, не стоит ударяться в панику. Все, что потребуется сделать для исправления положения, – это добавить нижеследующую строку в файл *~/.forward*:

```
|IFS=' ' && exec /usr/bin/procmail || exit 75 #myid
```

Здесь нужно лишь вместо строки `/usr/bin/procmail` подставить корректный путь к исполняемому файлу Procmail, а вместо `myid` – свой числовой идентификатор пользователя. (Это необходимо во избежание проблем с почтовыми клиентами, пытающимися оптимизировать доставку почты.)

Теперь можно попробовать послать письмо себе самому еще раз и убедиться, что все работает. Если файл так и не появился, это может быть следствием чересчур

закрытой системы. Проверьте права доступа к файлам *.procmailrc* и *.forward* – они должны быть доступны для чтения всем, а для записи (желательно) только вам. Вероятно, придется также добавить признак *x* к атрибутам домашнего каталога, что можно сделать следующей командой:

```
chmod go+x ~/
```

Если Procmail по-прежнему не запускается, самое время кричать «караул» или, по крайней мере, проконсультироваться по этому вопросу у производителя дистрибутива.

Настройка «песочницы»

Готовы ли вы потерять всю свою почту, играя с настройками Procmail? Если нет, тогда самое время настроить «песочницу» для проведения тестов. Создайте каталог *Test*, скопируйте туда файл *.procmailrc* и переименуйте его в *proctest.rc*. Теперь в процессе тестирования редактироваться будет этот файл, а не настоящий *.procmailrc*. В каталоге *Test* создайте сценарий со следующим содержимым:

```
#!/bin/sh
#Исполняемый файл с именем "proctest"
#
# Тестовый каталог.
TESTDIR=~/Test
if [ ! -d ${TESTDIR} ] ; then
    echo "Directory ${TESTDIR} does not exist; First create it"
    exit 0
fi

procmail ${TESTDIR}/proctest.rc < mail.msg
```

Желательно было бы изменить строку `LOGFILE` в файле *proctest.rc*, чтобы отладочные записи попадали не в реальный файл журнала, а в журнал, находящийся в каталоге *Test*. Кроме того, чтобы увеличить детализацию выводимых в журнал сообщений, можно добавить следующие строки:

```
VERBOSE=yes
LOGABSTRACT=all
```

Наконец, для проведения тестов необходимо поместить сообщение электронной почты в файл *mail.msg* и запустить сценарий *proctest*. Большинство программ электронной почты сохраняют сообщения в виде отдельных файлов. Если у вас еще нет ни одного сообщения, пошлите тестовое письмо себе самому и скопируйте файл */var/spool/mail/имя-пользователя*.

Синтаксис рецептов

После выполнения подготовительных операций можно перейти к рассмотрению рецептов. Все рецепты оформляются следующим образом:

```
:0 [флаги] [ : [локальный-файл-блокировки] ]
  <0 или более условий (по одному в строке)>
  <единственная строка, описывающая действие>
```

Строки с условиями начинаются с символа `*`. Все, что находится вслед за этим символом, передается внутренней функции *egrep* без изменений, за исключением пробелов в начале и в конце строки.

Строка, описывающая действие, может иметь разные формы:

- Если строка начинается с символа !, остальная часть строки будет восприниматься как адрес электронной почты, куда следует перенаправить сообщение.
- Если строка начинается с символа |, остальная часть строки будет восприниматься как команда оболочки, которую необходимо выполнить.
- Если строка начинается с символа {, все, что будет встречено до символа }, будет рассматриваться как вложенный блок. Вложенные блоки могут содержать несколько рецептов.
- Все остальное рассматривается как имя почтового ящика.

Флаги – это комбинация односимвольных параметров. Описание флагов приводится в табл. 23.1 (это описание взято со страниц справочного руководства команды `procmailrc`). Пока не стоит углубляться в изучение этой таблицы, просто вернитесь к ней после того, как будут рассмотрены примеры в следующем разделе.

Таблица 23.1. Флаги Procmail

Флаг	Назначение
`H`	Выполнить расширенный поиск в заголовке с помощью регулярного выражения (по умолчанию).
`B`	Выполнить расширенный поиск в теле письма с помощью регулярного выражения.
`D`	Выполнить поиск с помощью регулярного выражения с учетом регистра символов (по умолчанию регистр символов не учитывается.
`A`	Выполнить рецепт, только если не было выполнено условие самого последнего рецепта без флага `A` или `a` в текущем вложенном блоке.
`a`	То же, что и `A`, только в этом случае предыдущий рецепт должен был завершиться успехом.
`E`	Выполнить рецепт, только если не был выполнен предыдущий рецепт. Выполнение данного рецепта запрещает выполнение всех последующих с флагом `E`. Этот флаг дает возможность оформлять ветки `else if` в рецептах.
`e`	Выполнить рецепт, только если предыдущий рецепт был выполнен частично.
`h`	Записать содержимое заголовка в канал, файл или отправить получателю (по умолчанию).
`b`	Записать содержимое тела письма в канал, файл или отправить получателю (по умолчанию).
`c`	Создать копию сообщения для обработки последующими рецептами или для доставки получателю.
`w`	Дождаться завершения программы обработки и проверить возвращаемое ею значение.

Флаг	Назначение
W	То же, что и w, но с подавлением сообщений об ошибках, получаемых от программы.
i	Игнорировать ошибки записи (обычно при попытке записи в закрытый канал).
r	Режим неформатированных данных. Не гарантирует, что сообщение будет заканчиваться пустой строкой.

Строка условия – это, как правило, регулярное выражение, с помощью которого производится поиск в заголовке или в теле письма. Регулярные выражения описывались в главе 19. Однако в некоторых случаях могут использоваться специальные типы условий. Такие условия должны начинаться с одного из символов, которые приводятся в табл. 23.2.

Таблица 23.2. Флаги условий Procmail

Условие	Назначение
!	Выполнить действие, только если условие не выполняется.
$	Интерпретировать текст в двойных кавычках в соответствии с правилами подстановки, используемыми командной оболочкой bash.
?	Представляет значение, возвращаемое указанной программой.
<	Выполнить рецепт, если общая длина сообщения в байтах меньше заданного числа.
>	Выполнить рецепт, если общая длина сообщения в байтах больше заданного числа.
переменная	Сравнить последующий текст со значением указанной переменной, которая может быть переменной окружения или специальной переменной, имя которой представляет комбинация символов B и H, где B означает тело (body) письма, а H – заголовок (header).
\	Экранировать (воспринимать как простой символ) символы из этой таблицы, если они должны находиться в начале строки и интерпретироваться как простые символы.

Примеры

Разобраться с рецептами Procmail будет гораздо проще на примерах, поэтому в остальной части раздела будут продемонстрированы типичные примеры рецептов. Дополнительные примеры можно найти на страницах справочного руководства к procmailex.

Все примеры, следующие ниже, не являются полными сценариями Procmail, поэтому вам придется вернуться к разделу «Подготовка Procmail к использованию» и вспомнить правила оформления сценариев.

И последнее замечание: в процессе исследования рецептов не забывайте, что Procmail обрабатывает их в определенном порядке. Так, если рецепт пометит

почтовое отправление как доставленное, оно не будет обрабатываться последующими рецептами.

Создание резервной копии входящей почты

При изучении Procmail существует вероятность создать такой рецепт, который уничтожит почтовые сообщения по ошибке. Поэтому будет далеко не лишним воспользоваться следующим ниже рецептом, который должен стоять самым первым в настройках Procmail:

```
:0c:
backup
```

В первой строке рецепта стоит флаг `c`, который требует, чтобы почтовое сообщение было передано для обработки в других рецептах, даже если условия данного рецепта будут выполнены (если условия в рецепте отсутствуют, как в данном случае, считается, что сообщение соответствует этим условиям).

Вслед за флагом стоит двоеточие, которое указывает, что рецепт должен использовать локальную блокировку. То есть, прежде чем будет выполнено действие, предписываемое рецептом, должна быть получена блокировка, которая будет удерживаться в течение выполнения действия.

Заключительная часть рецепта – строка `backup`, которая воспринимается как имя почтового ящика. Если *$MAIL/backup* – это имя каталога (известного также как *maildir*), тогда письмо будет записано в этот каталог в виде файла с уникальным именем. Если *$MAIL/backup* – это имя файла, тогда письмо будет дописано в конец этого файла (известного также как *mbox*).

Сохранение почтовых рассылок в отдельном ящике

Следующий рецепт очень часто используется в Procmail: он сохраняет почтовые отправления, полученные из списков рассылки, в отдельном почтовом ящике. Выглядеть такой рецепт может следующим образом:

```
:0:
* Return-Path:.*kde-devel-bounces
kde-devel
```

Обратите внимание на отсутствие флага `c`. На этот раз цель рецепта – переместить сообщения из указанного списка рассылки в почтовый ящик *kde-devel* и не передавать его в ящик для входящей почты.

Строка, начинающаяся с символа звездочки, – это условие, при выполнении которого срабатывает рецепт. Эта строка представляет собой регулярное выражение, говорящее, что текст в заголовке письма должен содержать строку `Return-Path:`, вслед за которой может следовать любой текст (регулярное выражение `.*`) и далее должен находиться текст `kde-devel-bounces`. Мысль о том, как построить такое регулярное выражение, пришла к нам после ознакомления с почтовыми сообщениями, пришедшими из списка рассылки. Отыскать такое регулярное выражение, которое однозначно идентифицировало бы почту, получаемую из списка рассылки, и давало бы отрицательные результаты для всех остальных отправлений, всегда было непростым делом.

Перенаправление сообщений через SMS

Следующий рецепт перенаправляет почтовые сообщения, тема которых начинается с текста SMS, на мобильный телефон в виде текстовых сообщений, через воображаемый шлюз.

```
:0
* < 1000
* Subject: SMS
! 12345678@smsgateway.com
```

Данный рецепт содержит два условия: первое ограничивает размер письма, которое может быть перенаправлено, величиной в 1000 байт, а второе проверяет, начинается ли тема письма со строки `SMS`.

Строка описания действия начинается с символа восклицательного знака, таким образом, все, что следует дальше в этой строке, воспринимается как адрес электронной почты, куда следует перенаправить письмо.

Автоматическое создание ответов

В заключительном примере мы покажем, как автоматически создавать ответы на почтовые отправления. Во многих системах для этих целей используется программа с именем *vacation*, однако наш пример обеспечивает более высокую гибкость, так что, когда вы обретете определенные навыки разработки сценариев, вы без труда сможете изменить его и приспособить для своих нужд. Данный рецепт выглядит следующим образом:

```
:0c
* !^FROM_DAEMON
* !^X-Loop: your@own.mail.address
{
    SUBJECT=`formail -zx subject:`

    :0
    | (formail -r -I"Precedence: junk" \
    -A"X-Loop: your@own.mail.address" ; \
    echo "Ваше письмо с темой \"$SUBJECT\" было доставлено."; \
    echo "Однако сейчас меня нет на месте, но я отвечу как только смогу ") |
$SENDMAIL -t
}
```

Опять начнем с условий. Этот рецепт создает и отправляет ответ, только если, во-первых, почта получена не от демона почты и, во-вторых, почта не содержит в заголовке строку `X-Loop: your@own.mail.address` (конечно же, здесь должен находиться фактический адрес вашей электронной почты). Первое условие гарантирует, что не будет отправлено сообщение в ответ на письмо, полученное из списка рассылки, и второе условие предотвращает возможность зацикливания при получении аналогичного сообщения, сгенерированного автоматически в какой-нибудь другой системе.

Действие, которое выполняется при соблюдении обоих условий, представляет собой блок рецептов. Независимо от того, что заключено в фигурные скобки, этот блок будет интерпретироваться как обычный сценарий Procmail. Если исполне-

ние сценария достигнет конца блока (то есть сообщение не будет помечено как доставленное), оно будет продолжено за пределами блока. Но это не наш случай.

Первая строка блока – это операция присваивания значения переменной SUBJECT, которое является результатом исполнения команды *formail*. Эта утилита входит в состав пакета Procmail, она предназначена для обработки электронных писем или извлечения отдельных частей из них.

Вторая часть блока описывает выполняемое действие. Здесь составляется ответ, который затем отправляется по обратному адресу тому, кто послал вам электронное письмо. Рассмотрим эту часть по порядку.

Сначала вызывается команда *formail -r*, с помощью которой создается заголовок ответа из заголовка входящего сообщения. Эта команда отбросит все ненужные для ответа поля в исходном заголовке. Кроме того, команде передаются два дополнительных параметра командной строки: `-I"Precedence: junk"` и `-A "X-Loop: your@own.mail.address"`. Эти два параметра добавляют новые заголовки к письму: первый сообщает об уровне приоритета почты, а второй добавляет строку, аналогичную той, что проверяется одним из наших условий во избежание зацикливания.

До сих пор создавался заголовок ответа на письмо, который выводился на устройство стандартного вывода. Теперь с помощью команд *echo* на устройство стандартного вывода отправляется собственно текст письма-ответа. В заключение письмо целиком передается команде *sendmail*. Ключ *-t* указывает *sendmail* на необходимость определения адреса получателя из самого письма.[1]

Фильтрация спама

Постоянно увеличивающийся поток так называемого спама (более точно – непрошенная электронная почта рекламного характера) значительно снижает значимость электронной почты как среды общения. К счастью, существуют инструментальные средства, которые помогут справиться с этой проблемой. Называются они *фильтрами спама*, и главное их предназначение – классифицировать каждое входящее сообщение в соответствии с большим набором правил, чтобы определить, не является ли оно спамом. После этого фильтры или помечают сообщение дополнительными строками в заголовке, или изменяют строку темы. Затем вы (или программа – клиент электронной почты) должны рассортировать сообщения согласно некоторым критериям по отдельным папкам (или сразу в «корзину», что грозит потерей отправлений, классифицированных неверно). В конце рабочего дня можно будет решить, насколько агрессивно следует обрабатывать спам. Здесь важно определиться, что для вас более важно: отфильтровывать как можно больше спама или гарантировать, что ни одно важное сообщение (например, запрос потенциального клиента) не будет отфильтровано.

Существуют два различных способа фильтрования спама: непосредственно на сервере электронной почты или почтовым клиентом. Предпочтительнее выполнять фильтрацию на сервере, если почтовый сервер обслуживает более чем один клиент, потому что для всех пользователей данного почтового сервера может ис-

[1] При вызове с ключом *-t* утилита *sendmail* будет брать адрес отправителя не из командной строки, а из заголовка письма (поля `To:`, `Cc:` и `Bcc:`). – *Примеч. перев.*

пользоваться и поддерживаться один и тот же набор правил фильтрации, и сообщение, поступающее сразу нескольким пользователям этого сервера, должно пройти фильтр спама всего один раз, что снижает объем времени, затрачиваемого на обработку. С другой стороны, фильтрация на стороне клиента позволяет определять свои собственные правила фильтрации и выбирать используемый фильтр спама.

Самый известный в мире Linux фильтр спама (причем он никак не зависит от Linux) – это инструмент под названием SpamAssassin. Всю необходимую информацию о SpamAssassin можно найти на домашней странице проекта *http://spamassassin.spache.org*. SpamAssassin может работать и на стороне сервера, и на стороне клиента. Мы оставляем вам для самостоятельного изучения вопрос установки SpamAssassin на почтовом сервере Postfix (или другом), в чем вам поможет документация на веб-сервере проекта.

Когда SpamAssassin работает на стороне сервера, наилучшие результаты достигаются при использовании его в режиме «клиент-сервер». Благодаря этому исключается необходимость повторного запуска SpamAssassin и считывания объемных таблиц для проверки каждого сообщения. Вместо этого SpamAssassin исполняется как демон с именем *spamd*, к которому обращаются для проверки каждого сообщения с помощью интерфейсной команды *spamc*.

Если предполагается настроить почтовый клиент на использование SpamAssassin, вам необходимо будет передать каждое входящее электронное письмо команде *spamassassin* (конечно же, у вас сохраняется возможность использовать связку *spamc*/*spamd* на стороне клиента). Команда *spamassassin* принимает входящие сообщения с устройства стандартного ввода, анализирует их и выводит измененные сообщения на устройство стандартного вывода. Наиболее современные почтовые клиенты обладают возможностью конвейерной пересылки всех (или некоторых) входящих сообщений через внешнюю команду, таким образом, вам необходимо будет определить способ взаимодействия с командой *spamassassin*.

Если SpamAssassin классифицировал сообщение как спам, в заголовок сообщения будет добавлена строка:

```
X-Spam-Status: Yes
```

После этого вам лишь останется настроить фильтры внутри клиента, которые будут обрабатывать такие сообщения требуемым для вас образом (складывать в отдельную папку, уничтожать и т. д.). При желании более тонкой фильтрации можно дополнительно анализировать строку заголовка, начинающуюся с символов:

```
X-Spam-Level:
```

Вслед за этим маркером следует последовательность звездочек – чем больше число звездочек, тем выше вероятность, что данное сообщение является спамом.

Прежде чем перейти к рассмотрению одного из почтовых клиентов, подведем некоторые итоги. Чтобы настроить SpamAssassin для работы с почтовым клиентом, необходимо выполнить следующее:

- Настроить почтовый клиент так, чтобы все входящие сообщения по конвейеру передавались команде *spamassassin*.
- Предусмотреть фильтрацию сообщений на основе строк заголовка, добавляемых SpamAssassin.

Для этих целей можно даже использовать команду *procmail* (о ней рассказывалось в предыдущем разделе), которая будет передавать сообщения *spamassassin*. Информацию о том, как это сделать, можно найти на сайте *http://wiki.apache.org/spamassassin/UsedViaProcmail*.

Настройку поддержки SpamAssassin в почтовом клиенте мы рассмотрим на примере KMail – клиенте электронной почты, входящем в состав окружения рабочего стола KDE. Программа KMail позволяет выполнить все только что упомянутые действия. Но кроме того, она позволяет автоматизировать процедуру настройки с помощью Мастера антиспама. Чтобы запустить мастер, необходимо выбрать пункт меню Tools (Сервис)→Anti-Spam Wizard (Мастер антиспама). Этот мастер сначала проверит наличие и состав инструментальных средств борьбы со спамом в системе (он попытается отыскать не только SpamAssassin) и затем предложит выбрать те, которые должны использоваться программой KMail. Не следует стремиться выбирать все имеющиеся средства, потому что каждый дополнительный фильтр замедляет обработку поступающих почтовых сообщений.

На следующей странице мастера будут предложены варианты обработки спама. Как минимум следует отметить пункты Classify messages using the anti-spam tools (Классифицировать сообщения с помощью программ определения спама) и Move detected messages to the selected folder (Перемещать 100% спам в). После этого нужно выбрать папку, куда будут помещаться сообщения, с абсолютной надежностью классифицированные как спам, а также папку для сообщений, которые были классифицированы как спам с меньшей степенью надежности. После щелчка на кнопке Finish (Готово) программа KMail настроит все необходимые правила фильтрации, и при последующем чтении почты вы сможете увидеть, как заполняются папки для спама. Спам все еще будет попадать в папку для входящих сообщений, но намного реже, чем раньше.

SpamAssassin обладает значительным числом функциональных возможностей, о которых мы здесь даже не упомянули. Например, этот инструмент включает в себя байесовский фильтр, работа которого построена на основе статистического анализа. Когда нежелательное сообщение не помечается как спам, можно обучить SpamAssassin распознавать подобные сообщения как спам. Аналогично, если сообщение ошибочно распознается как спам, можно обучить SpamAssassin не рассматривать такие сообщения как спам в будущем. Как это сделать, описывается в документации к SpamAssassin.

Мы обсудили множество параметров настройки почтовой системы. Наш совет: начинайте не спеша, настраивайте систему по частям и проверяйте работоспособность настроек после каждого шага. Выполнить все настройки одним махом – очень сложная задача.

24

Запуск сервера FTP

В этой главе будут рассмотрены действия, которые необходимо выполнить для настройки собственного сервера FTP. В частности, будет рассматриваться сервер *ProFTPD*, представляющий собой свободно распространяемую реализацию очень стабильного сервера FTP, обладающего весьма широкими возможностями.

Введение

ProFTPD – это сервер FTP, распространяемый на условиях лицензии GPL и обладающий широкими возможностями настройки. Этот сервер можно рассматривать как FTP-аналог веб-сервера Apache. Чтобы такие аналогии имели право на существование, сервер не может отличаться бедностью настроек и средними возможностями, и действительно, данный сервер отличается чрезвычайной гибкостью. Сервер ProFTPD, например, используется таким известным проектом, как SourceForge.

Получить этот сервер можно на сайте *http://www.proftpd.org*.

Сборка и установка

Если ProFTPD не входит в состав вашего дистрибутива, вы можете либо собрать и установить сервер из исходных текстов, либо воспользоваться пакетом, подходящим для вашего дистрибутива. Дистрибутивы в формате RPM можно установить с помощью менеджера пакетов с сайта *http://proftpd.org*. Пользователи Debian с этой целью могут выполнить команду *apt-get install proftpd*.

Установка пакета RPM

Если не удастся отыскать дистрибутив RPM с двоичными файлами для вашего дистрибутива, тогда можно загрузить пакет RPM с исходными текстами и собрать сервер командой *rpmbuild --rebuild proftpd-1.2.10-1.src.rpm*. В результате будут получены два пакета RPM, готовых к установке: *proftpd-1.2.10-1.i586.rpm*, внутри которого будет находиться скомпилированное программное обеспечение сервера, и *proftpd-inetd-1.2.10-1.i586.rpm* с файлами поддержки возможности запуска ProFTPD через *xinetd*. Пакет *proftpd-inetd* является необязательным, и мы

не будем рассматривать его в этой книге. Установить основной пакет можно следующей командой:

```
# rpm -ivh /usr/src/packages/RPMS/i586/proftpd-1.2.10-1.i586.rpm
```

Формат RPM в основном предназначается для распространения пакетов дистрибутива Red Hat, поэтому, если вы пользуетесь SUSE, придется выполнить некоторые дополнительные действия. При установке пакета *rc*-сценарий сохраняется в каталоге */etc/rc.d/init.d/proftpd*, что не годится для SUSE. Кроме того, сценарий сам по себе не работоспособен, поэтому его нужно заменить следующим ниже сценарием и сохранить его под именем */etc/rc.d/proftpd*:

```
#!/bin/sh
#
# Сценарий запуска ProFTPD
#
# chkconfig: 345 85 15
# описание: ProFTPD - расширенный FTP-сервер, основная цель которого - обеспечить \
#      простоту настроек и высокую степень безопасности. Синтаксис настроек очень \
#      близок к серверу Apache, а инфраструктура сервера отличается широтой \
#      возможных настроек, включая поддержку нескольких «виртуальных» FTP-серверов,\
#      анонимного FTP-сервера и доступ к каталогам на основе разрешений.
# имя процесса: proftpd
# файл с настройками: /etc/proftpd.conf

PROFTPD=/usr/sbin/proftpd
PATH="$PATH:/usr/sbin"

if [ -f /etc/sysconfig/proftpd ]; then
    . /etc/sysconfig/proftpd
fi

. /etc/rc.status
rc_reset

# Определить, как вызывался данный сценарий.
case "$1" in
    start)
        echo -n "Starting proftpd: "
        startproc $PROFTPD $OPTIONS
        rc_status -v
        ;;
    stop)
        echo -n "Shutting down proftpd: "
        killproc -TERM $PROFTPD
        rc_status -v
        ;;
    try-restart)
        $0 status
        if test $? = 0; then
            $0 restart
        else
            rc_reset # Не был запущен - это не ошибка.
        fi
        # Сохранить статус и ничего не выводить
        rc_status
        ;;
    status)
```

```
            checkproc $PROFTPD
            rc_status -v
            ;;
        restart)
            $0 stop
            $0 start
            rc_status
            ;;
        reload)
            echo -n "Re-reading proftpd config: "
            killproc -HUP $PROFTPD
            rc_status -v
            ;;
        suspend)
            hash ftpshut>/dev/null 2>&1
            if [ $? = 0 ]; then
                if [ $# -gt 1 ]; then
                    shift
                    echo -n "Suspending with '$*' "
                    ftpshut $*
                else
                    echo -n "Suspending NOW "
                    ftpshut now "Maintanance in progress"
                fi
            else
                echo -n "No way to suspend "
            fi
            echo
            ;;
        resume)
            if [ -f /etc/shutmsg ]; then
                echo -n "Allowing sessions again "
                rm -f /etc/shutmsg
            else
                echo -n "Was not suspended "
            fi
            echo
            ;;
    *)
        echo -n "Usage: $0 {start|stop|restart|try-restart|status|reload|resume"
        hash ftpshut
        if [ $? = 1 ]; then
            echo '}'
        else
        echo '|suspend}'
        echo 'suspend accepts additional arguments which are passed to ftpshut(8)'
        fi
        exit 1
    esac
    rc_exit
```

Конечно же, в более поздних версиях такое положение вещей может быть уже исправлено.

Сборка из исходных текстов

Необходимо загрузить файл дистрибутива, распаковать, настроить и собрать:

```
$ tar xfj proftpd-1.2.10.tar.bz2
$ cd proftpd-1.2.10
$ ./configure --prefix=/usr/local/packages/proftpd
$ make
```

После этого, обладая привилегиями суперпользователя, нужно выполнить команду *make install*. Данная команда установит программное обеспечение в каталог */usr/local/packages/proftpd*.

Запуск ProFTPD

Запуск сервера

После того как *rc*-сценарий будет переписан в надлежащий каталог, сервер может быть запущен командой */etc/rc.d/proftpd start* (в Debian: */etc/init.d/proftpd start*).

Останов сервера

Чтобы остановить работу демона FTP, нужно запустить команду */etc/rc.d/proftpd stop*.

Временная приостановка сервера

С помощью команды */etc/rc.d/proftpd suspend* можно заставить сервер ProFTPD остановить прием новых соединений. Пользователи, пытающиеся соединиться с сервером, будут получать сообщение, извещающее, что сервер находится на техническом обслуживании. Чтобы продолжить работу в обычном режиме, необходимо дать команду */etc/rc.d/proftpd resume*.

Отладка

Во время поиска неисправностей или при изменении файла с настройками бывает желательно получить дополнительные сведения о том, что происходит в недрах сервера. Команда *proftpd -vv* выводит информацию о версии, *proftpd -nodaemon* запустит сервер как обычный процесс переднего плана, а команда *proftpd -t* проверит синтаксис текущего файла с настройками. Степень подробности выводимой информации можно повысить, если запустить сервер ProFTPD командой *proftpd -d9*. Параметры *-d0*–*-d9* могут указываться в сочетании с другими параметрами командной строки.

Настройка

Введение

При установке как из RPM-пакета, так и из исходных текстов настройки по умолчанию предполагают возможность анонимного доступа к публичному каталогу только на чтение и полноценный доступ для пользователей системы. Эти настройки могут служить отправной точкой для тех, кто собирается предоставить анонимный доступ к своему FTP-серверу.

Настройки сервера ProFTPD хранятся в файле */etc/proftpd.conf* или *$prefix/etc/proftpd.conf*, если установка производилась из исходных текстов. Анонимные пользователи FTP автоматически получают доступ к домашнему каталогу поль-

зователя FTP в специальном режиме, когда этот каталог рассматривается как корневой каталог файловой системы (*chroot*-окружение).

Файл *proftpd.conf* состоит из набора директив. Описание всех директив можно найти на сайте *http://www.proftpd.org/docs/directives/configuration_full.html*. Файл делится на несколько разделов, или *контекстов*, каждый из которых описывает настройки отдельного аспекта ProFTPD.

Главный контекст сервера

Часть файла с настройками, которая не входит в состав какого-либо контекста. Используется для описания глобальных настроек сервера, которые обычно располагаются в самом начале файла.

<Anonymous>

Этот контекст используется для настройки анонимного доступа к FTP-серверу. По умолчанию ProFTPD позволяет анонимный доступ к каталогу FTP без пароля и в *chroot*-окружении.

<Directory>

Этот контекст используется для настройки доступа к отдельным каталогам. Обычно используется для предоставления или ограничения доступа.

<Limit>

Этот контекст используется для управления доступом к командам и группам команд FTP, в зависимости от того, кто из пользователей пытается их выполнять.

<Global>

Этот контекст используется для описания виртуальных серверов (то есть когда ProFTPD обслуживает несколько сетевых интерфейсов, причем каждый из них может иметь различные параметры настройки). Директивы в этом контексте имеют то же назначение, что и директивы главного контекста сервера, за исключением того, что они могут быть переопределены в любом из контекстов `<VirtualHost>`.

<VirtualHost>

С помощью директивы `<VirtualHost>` можно создавать независимые наборы параметров настройки для различных сетевых интерфейсов и портов.

В следующем разделе представлены два примера настройки ProFTPD: типовой сервер FTP и улучшенный, где ProFTPD использует для организации прав доступа свою собственную базу данных с пользователями.

Типовые настройки

В данном примере настроек предоставляется возможность анонимного доступа к публичному разделу FTP-сервера и доступ ко всей файловой системе для зарегистрированных пользователей:

```
ServerName      "ProFTPD Default Installation"
ServerType      standalone
```

Директива `ServerName` определяет имя сервера, которое видят пользователи при подключении к серверу. Директива `ServerType` может принимать значения `stand-`

alone или inetd и определяет, будет ли ProFTPD самостоятельно принимать запросы на подключение или запускаться с помощью *(x)inetd*.

```
DefaultServer     on
Port              21
```

Директива DefaultServer on означает, что параметры настройки применимы ко всем сетевым интерфейсам хоста, а директива Port определяет номер порта, на котором ProFTPD будет принимать соединения (порт с номером 21 – это стандартный порт службы FTP):

```
Umask                 022
MaxInstances          30
User                  nobody
Group                 nogroup
AllowOverwrite        on
<Limit SITE_CHMOD>
DenyAll
</Limit>
```

Директива Umask эквивалентна команде *umask* командной оболочки. Директива MaxInstances устанавливает верхний предел параллельно исполняющихся дочерних процессов ProFTPD – то есть одновременно к серверу могут быть подключены не более 30 пользователей. Директивы User и Group определяют имя пользователя и группы, с привилегиями которых ProFTPD выполняет непривилегированные операции; привилегированные операции выполняются с правами аутентифицированного пользователя. Директива AllowOverwrite on определяет допустимость перезаписи файлов, доступных для записи. Раздел <Limit> блокирует возможность запуска команды *site chmod*.

```
<Anonymous ~ftp>
User                  ftp
Group                 ftp
UserAlias             anonymous ftp
MaxClients            10
DisplayLogin          welcome.msg
DisplayFirstChdir     .message
<Limit WRITE>
DenyAll
</Limit>
</Anonymous>
```

Данная часть настроек описывает анонимный доступ к домашнему каталогу пользователя FTP в режиме только для чтения (нередко это каталог */srv/ftp*). Доступ к каталогу осуществляется с привилегиями пользователя *ftp*, причем одновременно могут быть подключены не более 10 пользователей. Директива DisplayLogin welcome.msg описывает имя файла, откуда будет браться текст приглашения к регистрации, а директива DisplayFirstChdir .message предписывает выводить содержимое файла *.message*, находящегося в текущем каталоге, когда пользователь впервые входит в него с помощью команды *cds*.

Расширенные настройки

В этом разделе рассматривается более сложный пример настройки, когда сведения о пользователях, которым разрешен доступ к серверу FTP, хранятся не

в обычной для UNIX базе данных пользователей, а в файле *passwd*, предназначенном исключительно для нужд ProFTPD. Кроме того, в данной конфигурации предоставляется ограниченный анонимный доступ к серверу.

Содержимое файла *proftpd.conf* выглядит следующим образом:

```
ServerName                  "Acme ftp server"
ServerType                  standalone
DefaultServer               on
ServerIdent on              "FTP Server ready."
UseReverseDNS               off
IdentLookups                off
DeferWelcome                on
Port                        21
MaxInstances                30
User                        ftp
Group                       nogroup
Umask                       022

<Limit LOGIN>
    Order Deny,Allow
    AllowGroup ftpusers
</Limit>

AuthPAM                     off
AuthUserFile                /etc/proftpd.passwd
AuthGroupFile               /etc/proftpd.group
RequireValidShell           off
DefaultRoot                 ~
DirFakeUser on              ~
DirFakeGroup on             ~

DisplayLogin                welcome.msg
DisplayFirstChdir           .message
TransferLog                 /var/log/xferlog
ScoreboardFile              /var/lib/proftpd/scoreFile

<Directory />
    AllowOverwrite          on
</Directory>

<Anonymous /srv/ftp/anonymous>
    User                    ftp
    Group                   ftp
    # Нам нужно, чтобы анонимные пользователи обладали привилегиями пользователя "ftp"
    UserAlias               anonymous ftp
    # Ограничить максимальное число анонимных пользователей, подключенных одновременно
    MaxClients              15
    <Limit LOGIN>
       AllowAll
    </Limit>
    # Ограничить возможность записи для анонимных пользователей
    <Limit WRITE>
        DenyAll
    </Limit>
    TransferRate RETR 40.0:1024
</Anonymous>
```

```
<Directory /srv/ftp/joe/upload>
    <Limit WRITE STOR DEL>
        AllowAll
    </Limit>
</Directory>
```

Для начала посмотрим, как обслуживаются пользователи. FTP – это старейший протокол, который передает пароли в открытом, незашифрованном виде, поэтому весьма желательно отделить пользователей FTP-сервера от «реальных» учетных записей системы. Для этого используются две директивы

```
AuthUserFile /etc/proftpd.passwd
AuthGroupFile /etc/proftpd.group
```

с помощью которых мы указываем ProFTPD, что следует использовать альтернативные файлы *passwd* и *group*. Эти файлы имеют тот же формат, что и стандартные файлы */etc/passwd* и */etc/group*. В качестве примера приведем содержимое файла */etc/proftpd.passwd*, который был создан исключительно в демонстрационных целях:

```
joe:$1$KdLsLL1G$LNGq21xp9l/4vhF/l/ON1.:20000:20000:Joe User:/srv/ftp/joe:
```

Данная учетная запись была создана с паролем «qwerty», который был хеширован с помощью утилиты *ftpasswd*. Эту утилиту можно найти в каталоге *contrib* дистрибутива ProFTPD. Файл */etc/proftpd.group* содержит единственную запись: `ftpusers:x:20000:`. Данная группа используется в разделе

```
<Limit LOGIN>
Order Deny,Allow
AllowGroup ftpusers
</Limit>
```

который блокирует возможность регистрации на FTP для обычных пользователей системы и позволяет регистрироваться только членам специальной группы *ftpusers*. Обратите внимание: это не то же самое, что файл */etc/ftpusers*, который используется для хранения списка пользователей системы, кому *не* позволено пользоваться сервером FTP. В документации говорится, что файл, имя которого определено в директиве `AuthUserFile`, *замещает* системный файл */etc/passwd*, но похоже, что в настоящее время это не совсем так, поэтому, чтобы разрешить авторизованный доступ только тем пользователям, что перечислены в альтернативном файле *passwd*, была создана отдельная группа.

Существует возможность определить несколько пользователей в файле */etc/proftpd.passwd* с одинаковыми числовыми идентификаторами. Это удобно, потому что дает возможность предоставлять доступ большому числу пользователей, не боясь исчерпать диапазон допустимых значений числовых идентификаторов пользователей. Чтобы обеспечить владение файлами для текущего аутентифицированного пользователя и группы, мы поместили директивы:

```
DirFakeUser on ~
DirFakeGroup on ~
```

Это лишь косметическое украшение, дающее пользователям ощущение, что они являются действительными владельцами файлов. Директива `ScoreBoard` используется для указания местоположения файла с информацией о текущем состоя-

нии сеанса. Данный файл необходим для нормальной работы таких утилит, как *ftpwho* и *ftpcount*. На этом мы заканчиваем рассмотрение главного контекста файла с настройками.

Следующий раздел файла – это контекст <Anonymous>, который описывает порядок доступа пользователей *anonymous* и *ftp* к каталогу */srv/ftp/anonymous* и ограничивает максимальное количество одновременно подключенных анонимных пользователей числом 15. Здесь же ограничивается пропускная способность сервера, в директиве TransferRate RETR 40.0:1024. Эти числа означают, что скорость загрузки ограничена величиной 40 Кбайт/с для файлов, чья длина превышает 1 Кбайт.

Последний контекст в файле с настройками описывает каталог */upload*, доступный для записи пользователю *joe*. По умолчанию ни один из каталогов не доступен ни одному пользователю, благодаря директиве <Limit WRITE> в главном контексте сервера. Таким образом, здесь пользователю *joe* выдаются специальные привилегии на возможность записи файлов в каталог */upload*.

Виртуальные хосты

Сервер ProFTPD поддерживает возможность создания виртуальных серверов в контексте <VirtualHost>. К сожалению, протокол FTP не поддерживает возможность виртуального хостинга, в отличие, например, от протокола HTTP, однако имеется возможность обслуживать различные порты или сетевые интерфейсы с разными параметрами настройки. Разумеется, все эти возможности доступны только в случае, если ProFTPD работает как автономный сервер, запускается с помощью *inetd* и все порты и интерфейсы находятся в руках *inetd*, а не ProFTPD.

Рассмотрим пример настройки нескольких виртуальных серверов:

```
ServerName                  "Acme FTP Server"
ServerType                  standalone

### Главный контекст сервера
# Установка привилегий, с которыми работает сервер в обычном режиме.
User nobody
Group nogroup
MaxInstances 30

# Контекст Global, где определяются «глобальные» параметры, которые будут
# использоваться и главным сервером, и всеми виртуальными хостами.

<Global>
    # Umask 022 - типичная маска, не позволяющая делать новые файлы
    # и каталоги доступными для записи группе и всем остальным.
    Umask 022
</Global>

### Виртуальный сервер, обслуживающий внутренний интерфейс
<VirtualHost 127.0.0.1>
    ServerName              "Acme Internal FTP"
    MaxClients              10
    DeferWelcome            on
    <Limit LOGIN>
        DenyAll
    </Limit>
```

```
    <Anonymous /srv/ftp/anonymous-internal>
        User                ftp
        Group               ftp
        AnonRequirePassword  off
        # Нам нужно, чтобы клиенты имели возможность зарегистрироваться
        # под именем "anonymous" и "ftp"
        UserAlias           anonymous ftp
        <Limit LOGIN>
            AllowAll
        </Limit>
        # Ограничить возможность записи при анонимном доступе
        <Limit WRITE>
            DenyAll
        </Limit>
    </Anonymous>
</VirtualHost>

### Еще один виртуальный сервер, обслуживающий порт с номером 4000
<VirtualHost 192.168.1.5>
    ServerName              "Acme Internal FTP upload"
    Port                    4000
    MaxClients              10
    MaxLoginAttempts        1
    DeferWelcome            on
    <Limit LOGIN>
        DenyAll
    </Limit>
    <Anonymous /srv/ftp/anonymous-upload>
        User                ftp
        Group               ftp
        AnonRequirePassword  off
        # Нам нужно, чтобы клиенты имели возможность зарегистрироваться
        # под именем "anonymous" и "ftp"
        UserAlias           anonymous ftp
        <Limit LOGIN>
            AllowAll
        </Limit>
        # Разрешается только запись файлов
        <Limit STOR CWD XCWD>
            AllowAll
        </Limit>
        <Limit READ DELE MKD RMD XMKD XRMD>
            DenyAll
        </Limit>
    </Anonymous>
</VirtualHost>
```

Этот пример содержит типичные настройки, допускающие возможность доступа ко всей файловой системе для пользователей UNIX. Наиболее интересной особенностью являются два раздела `<VirtualHost>`. Первый описывает исключительно анонимный сервер, прослушивающий локальный интерфейс (127.0.0.1) (что, на мой взгляд, не имеет большого смысла), а второй описывает анонимный сервер, прослушивающий порт 4000 на интерфейсе 192.168.1.5 и доступный исключительно для записи.

25

Запуск веб-приложений с использованием MySQL и PHP

Для большинства веб-сайтов недостаточно написать пару строк кода HTML; динамический контент – вот что сегодня востребовано. По правде говоря, большинство коммерческих сайтов предлагают такое динамическое наполнение своих страниц, которое в действительности не нужно посетителям, например мультипликационные ролики, не несущие никакой полезной информации, или интерактивные меню, созданные с помощью JavaScript, которые только усложняют доступ к информации вместо того, чтобы облегчать его. Однако в этой главе мы будем рассматривать динамическое наполнение совсем иного рода, которое действительно несет в себе полезную информацию.

Linux, как можно догадаться, – отличная платформа для генерации динамического контента. Несметное число веб-сайтов, предоставляющих динамический контент, уже работают под Linux; это одна из главных областей приложений, в которых отличается Linux.

Динамический контент можно получить, применяя два совершенно разных способа программирования: программирование на стороне сервера и программирование на стороне клиента. В программировании на стороне клиента интерактивные HTML-страницы чаще всего получают с помощью JavaScript, апплетов Java и платформы ActiveX от компании Microsoft.

Из-за ограничений, присущих этим технологиям, в большинстве случаев применяется программирование на стороне сервера. Есть много способов использования программ, выполняемых на сервере, и много различных программных пакетов для поддержки этой технологии, но существует одна комбинация таких пакетов, которая распространена повсеместно. Она стала настолько популярной, что для нее был придуман акроним LAMP, за которым скрывается Linux-Apache-MySQL-PHP. О веб-сервере Apache мы уже рассказывали, Linux посвящена книга в целом, поэтому нам осталось обсудить два последних пакета – MySQL и PHP, а также совместное применение всех четырех составляющих.

Чтобы получить действующую комбинацию LAMP, нужно настроить Apache, что подробно описано в разделе «Настройка собственного веб-сервера» главы 22,

а также установить MySQL и PHP. О том, как запустить в работу последние два пакета, будет рассказано в этой главе.

Но прежде чем перейти к техническим деталям, хочется обсудить, стоит ли труда установка и освоение системы LAMP.

LAMP позволяет легко обеспечить большой объем контента и навигацию по нему пользователей вашего веб-сайта.

Допустим, у вас есть сайт с множеством JPEG-файлов фотографий, которые вы сняли в разное время. Для посетителей сайта желательно иметь возможность выбрать различный размер просматриваемых фотографий. Одни захотят посмотреть фотографии архитектурных памятников, когда бы они ни были созданы; другие пожелают увидеть фотографии, сделанные во время вашего последнего путешествия, когда бы оно ни происходило.

Чтобы упростить навигацию и извлечение материала, вы сначала помещаете свои JPEG-файлы в базу данных MySQL. (Фактически файлы JPEG остаются существовать в файловой системе, откуда они быстро могут извлекаться веб-сервером Apache.) Вы организуете их по любому принципу (по теме, по путешествию и т. д.) и записываете всю информацию в таблицы базы данных. Другими словами, данные хранятся в виде отдельных таблиц, а комбинация взаимосвязанных таблиц составляет базу данных.

Теперь вы создаете форму, в которой посетители веб-сайта могут указать желаемые размеры фото. Это может быть простая форма, как показано на рис. 25.1.

Следующая ваша страница – динамическая, которая построчно будет описана в этой главе. Короткая программа на языке PHP получает запрос посетителя и определяет, что нужно показать на этой странице. Она может выглядеть, как показано на рис. 25.2.

А где же тут участвует MySQL? Ее участие не сразу видно, но она играет ключевую роль, поскольку программный код PHP запрашивает у нее данные. Сочета-

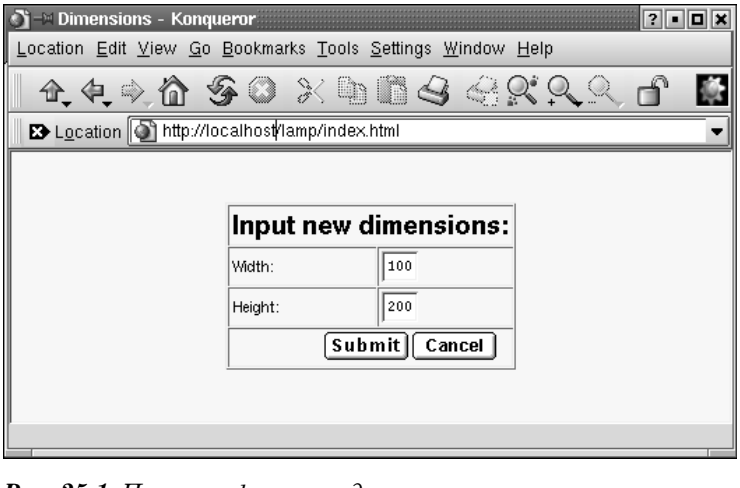


Рис. 25.1. *Простая форма ввода*

Рис. 25.2. *Динамическая веб-страница, созданная с помощью PHP*

ние встроенного кода PHP и быстрой базы данных оставляет у посетителя довольно приятное впечатление.

MySQL

MySQL[1] – база данных с открытым исходным кодом (open source), очень быстрая и относительно просто управляемая. Если вам требуются более сложные возможности баз данных, такие как репликации или распределенные базы данных, либо вы предполагаете хранение гигабайтов данных, то нечто крупное типа Oracle может оказаться лучшим выбором, но для большинства задач прекрасно подходит MySQL (и фактически она быстро нагоняет конкурентов по своим возможностям). Порядок использования MySQL регламентируется на основе двух лицензий. Если вы пользуетесь GPL-версией, ваше приложение также должно распространяться на условиях лицензии GPL, в противном случае вам необходимо приобрести коммерческую лицензию.

Установка и начальная настройка MySQL

Скорее всего, в вашем дистрибутиве есть система MySQL, которую можно установить, но если вам нужно все самое лучшее и свежее, можно зайти на *http://*

[1] Поль Дюбуа «MySQL. Сборник рецептов». – Пер. с англ. – СПб: Символ-Плюс, 2005.

www.mysql.com/downloads и загрузить весь пакет. На данный момент последняя стабильная версия имеет номер 4.1.13. Недавно была стабилизирована версия 5.

Если планируется использовать MySQL для реальной работы, необходимо, чтобы система была построена на основе ядра версии 2.4 или выше.

В версиях MySQL, скомпилированных с помощью *gcc* 2.96, могут происходить неожиданные аварии. Данная версия *gcc* не является официальной стабильной версией, но есть по крайней мере один дистрибутив (Red Hat), в который, к сожалению, эта версия включена в качестве компилятора по умолчанию. Поэтому, если у вас происходят непонятные аварии сервера базы данных и вы пользуетесь *gcc* 2.96, попробуйте установить готовый двоичный дистрибутив или воспользоваться более стабильной версией компилятора, например 3.3.5.

Если вы хотите скомпилировать MySQL самостоятельно, нужно загрузить пакет с исходными текстами, распаковать его и установить с помощью команд:

```
owl$ ./configure --prefix=/usr/local/mysql
owl$ make
owl# make install
```

Обратите внимание: в вашей системе путь для установки может оказаться иным. Кроме того, для третьей команды потребуется, вероятно, зарегистрироваться как *root*. Запомните путь установки, потому что позднее он потребуется для настройки PHP.

Следующим шагом мы рекомендуем создать пользователя и группу с именем *mysql*, как описано в разделе «Создание учетных записей» главы 11. Получите привилегии этого пользователя с помощью команды *su - mysql* и выполните:

```
owl$ scripts/mysql_install_db
```

О выборе баз данных

Наверное, следует отметить, что MySQL не единственный вариант выбора базы данных в качестве сервера для динамического веб-сайта. Например, Postgres[1], устанавливаемая по умолчанию на системах Red Hat, – тоже очень хорошая свободно распространяемая база данных. Поэтому при желании развернуть систему «LAPP», выбрав Postgres, вполне можно сделать это. Большая часть приводимой в данной главе информации будет применима и в такой конфигурации. Однако MySQL можно рассматривать как стандартную базу данных для динамических веб-сайтов на системах Linux. Это обусловлено простотой ее использования, скоростью и универсальностью. Кроме того, если вам потребуется дополнительная документация по развертыванию динамических веб-сайтов под Linux (к которой мы настоятельно рекомендуем обратиться), то более вероятно найти такую информацию (например, в виде специальных книг по этой теме) о MySQL, чем о Postgres.

---

[1] Р. Стоунз и Н. Мэттью «PostgreSQL. Основы». – Пер. с англ. – СПб: Символ-Плюс, 2002.

По соображениям безопасности полезно запретить пользователю *mysql* вход в систему. Это легко сделать, зарегистрировавшись как *root* и поместив звездочку во второе поле (пароль) записи в файле */etc/password* и/или */etc/shadow*.

После этого действия нужно выполнить еще одну команду как *root*, после чего можно вернуться к своей обычной учетной записи. Следующая команда запускает сервер MySQL:

```
owl# /usr/local/mysql/bin/safe_mysqld &
```

Можно также добавить параметр *--log* или *--log-long-format*, чтобы регистрировать в файле журнала все, что происходит с сервером базы данных.

Чтобы проверить, правильно ли запустился сервер, можно выполнить (в качестве обычного пользователя) следующую команду (если вы установили MySQL в другое место, нужно, конечно, изменить путь):

```
owl$ /usr/local/mysql/bin/mysqladmin version
mysqladmin Ver 8.41 Distrib 4.1.13, for suse-linux on i686
Copyright (C) 2000 MySQL AB & MySQL Finland AB & TCX DataKonsult AB
This software comes with ABSOLUTELY NO WARRANTY. This is free software,
and you are welcome to modify and redistribute it under the GPL license.

Server version                  4.1.13
Protocol version                10
Connection                      Localhost via UNIX socket
UNIX socket                     /var/lib/mysql/mysql.sock
Uptime:                         43 days 11 hours 39 min 17 sec
Threads: 1 Questions: 142 Slow queries: 0 Opens: 160 Flush tables: 1 Open
tables: 13 Queries per second avg: 0.000
```

Это должно работать без ввода пароля. Хотим отметить, однако, что работать с базой данных без пароля не рекомендуется, поскольку это увеличивает шансы потенциального злоумышленника получить доступ к ценным данным, которые могут находиться в вашей базе. Можно оставить базу данных без пароля на время тестирования, но завершив все проверки, нужно обязательно установить пароль и еще раз проверить, что все работает так же хорошо, когда пароль установлен. Если вы создали пароль для пользователя базы данных с именем *root* (либо это сделал за вас ваш дистрибутив; обращайтесь к документации в случае затруднений), нужно задать параметр *-p*, чтобы *mysqladmin* запросил ваш пароль.

Следует добавить, что в большинстве дистрибутивов есть сценарий для запуска сервера MySQL, которым можно воспользоваться вместо запуска сервера вручную (особенно если MySQL установлен с инсталляционного носителя). Часто этот сценарий помещается в файле */etc/init.d/mysql*.

Если сервер базы данных запущен и работает, можно приступить к созданию пользователей базы данных и новых баз данных. Следует отметить, что в исходные тексты MySQL включен вполне приличный учебник, кроме того, массу документации можно найти на *http://www.mysql.com*, поэтому мы расскажем лишь самое основное, что позволит быстро начать работу.

Начальные задачи: настройка учетных записей и SQL

Существуют три способа взаимодействия с ядром MySQL: можно пользоваться консольным клиентом, писать так называемые SQL-сценарии и посылать их базе данных, чтобы выполнять сразу несколько команд SQL, или воспользоваться одной из множества промежуточных библиотек, предоставляющих возможность обращаться к базам данных MySQL из программ, написанных на языке по вашему выбору (в зависимости от используемой библиотеки, может получиться так, что вам вообще не придется писать на языке SQL). Название SQL является сокращением от Structured Query Language – язык структурированных запросов, используемый для работы с реляционными базами данных. Далее в этой главе мы расскажем о его применении. Все три способа выполнения команд SQL предполагают, что вы правильно ввели имя пользователя и пароль.

Для работы с MySQL важно знать, что учетные записи пользователей Linux и пользователей MySQL различны. Иными словами, в MySQL есть собственная система управления бюджетами пользователей. Однако чаще всего пользователи дают своим учетным записям в MySQL такие же имена, как в Linux, чтобы избежать путаницы.

По умолчанию в MySQL есть одна учетная запись с именем *root*, для которой не установлен пароль (вот вам пример «защиты по умолчанию»...). Это означает, что можно получить доступ к серверу базы данных через интерактивную утилиту командной строки *mysql* следующим образом:

```
owl$ mysql -u root

Welcome to the MySQL monitor. Commands end with ; or \g.
Your MySQL connection id is 13 to server version: 4.1.13.

Type 'help;' or '\h' for help. Type '\c' to clear the buffer.

mysql>
```

Параметр *-u* позволяет указать пользователя базы данных. Если эта команда не работает, то, возможно, в вашем дистрибутиве MySQL для пользователя *root* установлен пароль. Найдите этот пароль в документации и запустите программу *mysql* снова:

```
owl$ mysql -u root -p
```

Вы должны получить приглашение для ввода пароля.

Допустим, что вы смогли подключиться к серверу базы данных. Тогда выполните команду[1]

```
mysql> show databases;
+-------------+
| Database    |
+-------------+
| mysql       |
| test        |
+-------------+
2 rows in set (0.11 sec)
```

[1] Это не команда SQL, а команда администрирования MySQL.

Она сообщает, что этот сервер управляет двумя базами данных. Одна из них называется *mysql* и содержит внутреннюю информацию MySQL, включая имена пользователей, а другая называется *test* и может использоваться для экспериментов. Создать новые базы данных очень просто, и мы покажем, как это делается, чуть позже. Как вы уже видели, все команды SQL должны оканчиваться точкой с запятой – вероятно, это должно доставить удовольствие программистам C.

Теперь нужно задать пароль для пользователя *root* (если его еще нет). Это делают две команды SQL:

```
mysql> SET PASSWORD FOR root=PASSWORD(new_topsecret_passwd');
mysql> FLUSH PRIVILEGES;
```

Снова обратите внимание на точку с запятой в конце этих команд: если вы забудете ввести этот символ и нажмете Enter, MySQL молча уставится на вас и будет ждать, когда вы введете что-нибудь еще.

Кстати, в командах SQL не учитывается регистр символов. Мы написали их заглавными буквами, потому что при этом легче определять в сценариях SQL, где находятся ключевые слова и переменные параметры.

Обратите также внимание на команду *FLUSH PRIVILEGES*. Она важна, поскольку только после ее выполнения MySQL обновляет свою базу данных пользователей.

Теперь мы хотим создать нового пользователя с именем *olof* с такими же правами доступа, как у *root*, за исключением права создания новых пользователей. За исключением этого права, у *olof* будут все привилегии для работы со всеми базами данных на этом сервере MySQL:

```
mysql> GRANT ALL PRIVILEGES ON *.* TO olof@localhost IDENTIFIED BY 'olof_passwd';
mysql> FLUSH PRIVILEGES;
```

Пользователь *olof* сможет подключаться к базе данных только с локальной машины. Это хорошее решение, не создающее лишних проблем с безопасностью. Мы советуем разрешать доступ только с локальной машины, если в обязательном порядке не требуется иного. Даже в связке LAMP достаточно локального доступа, потому что процесс веб-сервера выполняется на локальной машине и с базой данных соединяется этот процесс, а не процесс веб-броузера пользователя.

Но если вам действительно необходим доступ к базе данных по сети, можно воспользоваться такими командами:

```
mysql> GRANT ALL PRIVILEGES ON *.* TO username@"%" IDENTIFIED BY 'user_passwd';
mysql> FLUSH PRIVILEGES;
```

Если вам кажется чрезмерным предоставление всех прав доступа, за исключением создания новых пользователей, создадим еще одного пользователя, который сможет выполнять операции `SELECT`, `INSERT`, `UPDATE`, `DELETE` и `DROP`, но только в базе данных *test* (и только подключившись с локальной машины):

```
mysql> GRANT SELECT, INSERT, UPDATE, DELETE, DROP ON test.* TO gonzo@localhost
       IDENTIFIED BY 'gonzo_passwd';
mysql> FLUSH PRIVILEGES;
```

Если вам не доводилось работать с базами данных SQL прежде, эти операции могут показаться непонятными. Поскольку они все равно понадобятся вам при организации своей системы LAMP, мы кратко опишем их здесь:

`SELECT`

Наиболее часто используемая команда SQL. Она запрашивает в базе данных информацию с определенными свойствами. Например, можно запросить данные по всем клиентам в указанном городе. `SELECT` никак не модифицирует базу данных.

`INSERT`

Эта команда SQL вводит новые записи в таблицу базы данных. С ее помощью (в интерактивном режиме или чаще внутри некоторой программы) можно, например, ввести в таблицу клиентов базы данных запись о новом клиенте.

`UPDATE`

Эта команда SQL модифицирует записи, имеющиеся в базе данных. С ее помощью можно, например, увеличить розничные цены всех товаров, имеющихся в базе данных, на 15% (вот вам и инфляция!).

`DELETE`

Эта команда SQL удаляет из базы данных целые записи. Будьте осторожны с ней, поскольку восстановить данные будет нельзя, кроме как с резервной копии (если она есть!).

Есть и другие команды SQL и соответствующие привилегии (например, `DROP`, позволяющая удалять целые таблицы и даже целые базы данных), но они используются реже, чем приведенная «большая четверка».

Теперь мы хотим создать новую базу данных, в которую потом будем помещать таблицы и данные. Это делается командой SQL `CREATE DATABASE`:

```
mysql> create database test_database;
Query OK; 1 row affected (0.03 sec)
```

MySQL показала, что все прошло удачно, но для большей уверенности еще раз спросим, какими базами данных управляет сервер:

```
mysql> show databases;
+-----------------+
| Database        |
+-----------------+
| mysql           |
| test            |
| test_database   |
+-----------------+
6 rows in set (0.00 sec)
```

Создание и заполнение базы данных

Теперь мы хотим определить таблицу в нашей новой базе данных, но сначала необходимо сообщить серверу MySQL о том, что мы действительно хотим работать с этой базой данных:

```
mysql> use test_database
Database changed
```

Как видите, здесь мы не поставили в конце точки с запятой, поскольку это не команда SQL, а управляющий оператор для консольного клиента MySQL. Но не было бы ошибкой поставить и здесь точку с запятой.

Определение таблицы, где, в конечном счете, будут храниться ваши данные, происходит с помощью команды SQL `CREATE TABLE`, например:

```
mysql> CREATE TABLE comment_table(
    ->   id INT NOT NULL auto_increment,
    ->   comment TEXT,
    ->   PRIMARY KEY(id));
Query OK, 0 rows affected (0.10 sec)
```

Здесь мы определили таблицу `comment_table`, в которой есть две колонки, то есть два поля данных в каждой записи. Одна колонка называется *id* и содержит уникальный идентификатор каждой записи, который не может повторяться в различных записях, а потому помечен как *первичный ключ* (*primary key*), что на жаргоне баз данных означает «уникальный идентификатор». Другая колонка – это переменная типа `TEXT`, которая может содержать до 65 535 символов.

Теперь мы можем проверить, какие таблицы есть в нашей базе данных:

```
mysql> show tables;
+-------------------------+
| Tables_in_test_database |
+-------------------------+
| comment_table           |
+-------------------------+
1 row in set (0.00 sec)
```

Как видим, все в порядке, и можно начать добавлять в нашу таблицу записи с данными. Делается это с помощью SQL-команды `INSERT`:

```
mysql> INSERT INTO comment_table VALUES ('0','comment');
Query OK, 1 row affected (0.06 sec)
```

Наконец, можно проверить, какие данные содержатся в нашей таблице:

```
mysql> SELECT * FROM comment_table;
+----+---------+
| id | comment |
+----+---------+
| 1 | comment  |
+----+---------+
1 row in set (0.01 sec)
```

Мы запросили все (`*`) колонки таблицы `comment_table`. Возможно, вы заметили нечто странное: мы попросили MySQL вставить в первую колонку `0`, а вместо этого там оказалась `1`. Дело в том, что мы определили для этой колонки тип `INT NOT NULL auto_increment`, то есть значением колонки не может быть `NULL`, и MySQL должна автоматически выбрать следующее доступное значение. Это удобно, потому что можно вводить в таблицу новые записи, не беспокоясь о выборе уникальных значений для первой колонки:

```
mysql> INSERT INTO comment_table VALUES ('0','comment1');
Query OK, 1 row affected (0.00 sec)
```

```
mysql> SELECT * FROM comment_table;
+----+----------+
| id | comment  |
+----+----------+
| 1 | comment   |
| 2 | comment1  |
+----+----------+
2 rows in set (0.00 sec)
```

Как видите, мы снова задали 0 в качестве значения первой колонки, но MySQL автоматически выбрала очередное допустимое значение.

Теперь вы достаточно знаете о MySQL, чтобы приступить к самостоятельным экспериментам или начать читать какую-либо другую книгу о базах данных, мечтая создать очередной успешный веб-сайт электронной коммерции.

Но прежде чем погрузиться в сладкие грезы, позвольте нам закончить обсуждение MySQL и рассказать вам еще об одном замечательном ее средстве: сценариях SQL.

Не обязательно вводить все команды в приглашение командной строки клиента MySQL, можно выполнять пакетные файлы с командами SQL, направляя их на стандартный ввод программы *mysql*. Например, сохраним следующий код SQL как файл с именем *create_db.sql*:

```
DROP DATABASE IF EXISTS test_database;
CREATE DATABASE test_database;
USE test_database;
CREATE TABLE comment_table( id INT NOT NULL auto_increment,\
  comment TEXT,PRIMARY KEY(id));

INSERT INTO comment_table VALUES ('0','comment');
INSERT INTO comment_table VALUES ('0','comment1');
```

Выполнить этот сценарий можно из обычной командной строки Linux:

```
mysql -u root -p < create_db.sql
```

Строка

```
DROP DATABASE IF EXISTS test_database;
```

весьма опасна; ее можно использовать, только если в вашей базе данных нет важной информации.

По правде говоря, нет абсолютной необходимости создавать новую базу данных для каждого нового проекта, хотя это весьма рекомендуется. Теоретически можно сваливать все данные в базу данных *test*, которая устанавливается вместе с MySQL, если только все имена таблиц будут различными. На практике сопровождение такой базы данных превратится в кошмар, как только количество таблиц в ней станет значительным.

PHP

Для укомплектования нашего набора из Linux, Apache, PHP и MySQL нужен еще интерпретатор языка PHP. PHP – это рекурсивный акроним, раскрываемый как PHP: Hypertext Preprocessor. Он разрабатывается уже несколько лет. Наиболее распространенными версиями являются 4 и 5. В этой главе мы исполь-

зуем PHP4, поскольку на момент написания книги это самая распространенная стабильная версия.[1] Различия между версиями 4 и 5 либо скрыты в недрах реализации, либо это такие возможности, которые будут представлять интерес, когда у вас накопится богатый опыт работы с этим языком сценариев.

Примеры программного кода PHP

Приятной особенностью PHP является возможность ввода кода PHP непосредственно в код HTML. Веб-сервер передает все, что находится между тегами `<?php` и `?>`, модулю PHP, который интерпретирует и выполняет команды. Приведем очень простой пример кода PHP, помещенного на HTML-страницу; если вы уже установили PHP, можете выполнить его прямо из своего веб-сервера (если нет, ниже мы покажем, как устанавливать PHP):

```
<html>
<body>
<?php
echo "Hi,";
?>
LAMP enthusiasts.
</body>
</html>
```

Как вы, возможно, и предполагали, броузер покажет следующий текст:

```
Hi, LAMP enthusiasts.
```

Этот крайне простой пример показывает, как сервер Apache взаимодействует с интерпретатором PHP: код между `<?php` и `?>` передается интерпретатору PHP, который выполняет команду *echo*, а та выводит свои параметры в веб-броузер. Кроме того, строка `LAMP enthusiasts` просто добавляется в виде обычного текста HTML (и поскольку в ней нет разметки, она не похожа на HTML).

Конечно, возможности PHP значительно шире. Как и большинство языков программирования, он может использовать переменные и принимать решения, как в следующем сценарии (для краткости HTML-окружение опущено):

```
<?php
echo "Dear friends, today's date is: ";
echo date("F d, Y")."\n";
echo "<br>";
echo "We are in the ";

if ( date ("m") <= 6 ) {
    echo "first ";
} else {
    echo "second ";
}
echo "half of the year ".date("Y");
?>
```

[1] Л. Аргерих и др. «Профессиональное PHP программирование», 2-е издание. – Пер. с англ. – СПб.: Символ-Плюс, 2005.
Э. Гутманс, С. Баккен и Д. Ретанс «PHP 5. Профессиональное программирование». – Пер. с англ. – СПб.: Символ-Плюс, 2006.

Возможно, вы уже разобрались, что этот сценарий основывает свое решение в операторе `if` на том, какой сейчас месяц. Обратите внимание на тег HTML (`<br>`) в выводе PHP. Это совершенно допустимо и является распространенным приемом при работе с PHP. Ваш веб-броузер получит следующие данные (конечно, с другими датами, если только в вашем компьютере исправны часы и у вас не случилась деформация времени):

```
Dear friends, today's date is: May 04, 2002
<br>We are in the first half of the year of 2002
```

Веб-броузер сделает перенос в том месте, где стоит тег `<br>`.

Как и большинство языков программирования, PHP поддерживает функции в целях обеспечения модульности кода. Функции позволяют выполнять фрагмент кода в различных местах без многократного его дублирования.

PHP поставляется с весьма обширными библиотеками функций, которые можно дополнительно загружать из Интернета. Подключить библиотеку функций можно с помощью операторов `include()` и `require()`, имеющих между собой незначительные различия.

Если вы хотите программировать на PHP, следует ознакомиться с документацией по библиотекам функций, которые поставляются вместе с интерпретатором PHP, поскольку благодаря этим функциям вам не придется изобретать велосипед при решении стандартных задач.

Ниже приводится определение двух простых функций: `show_date`, которая выводит текущую дату в предопределенном формате и добавляет перевод строки, и `show_halfyear`, которая выводит `first` или `second` в зависимости от текущего месяца:

```
<?php

function show_date( ) {
    echo date("F d, Y") . "\n <br>";
}

function show_halfyear( ) {
    if (date("m") <= 6) {
        echo "first ";
    } else {
        echo "second ";
    }
}

?>
```

Назовем этот сценарий *functions.php* и перепишем наш первоначальный сценарий с использованием этих функций:

```
<?php

require("functions.php");
echo "Dear friends, the date today is: ";

show_date( );

echo "<br>";
```

```
echo "We are in the ";

show_semester( );

echo "semester of " . date("Y");
?>
```

Оператор `require()` сообщает интерпретатору PHP о необходимости загрузить наш сценарий с функциями и сделать содержащиеся в нем функции доступными текущему сценарию.

Конечно, мы очень поверхностно коснулись возможностей PHP. Если у вас появился к нему интерес, можете прочесть книгу «Programming PHP» Расмуса Лердорфа (Rasmus Lerdorf), создателя PHP, и Кевина Тэтроу (Kevin Tatroe), издательство O'Reilly.

До выхода PHP3 язык PHP был интерпретируемым, и его код хранился в буфере. Циклы и другие фрагменты кода, выполнение которых повторялось, многократно подвергались синтаксическому анализу – перед каждым выполнением кода. Конечно, достичь оптимальной производительности при этом было нельзя.

PHP4 был написан полностью заново и состоит из ядра языка (под названием «Zend») и функциональных модулей (которые очень гибки и расширяемы). В отличие от PHP3, PHP4 допускает работу в многопоточной среде, что позволяет использование PHP в качестве модуля различными веб-серверами. PHP5 также был переписан заново (в нем используется ядро под названием «Zend2») и обладает улучшенной, более интуитивно понятной моделью объектов, улучшенной производительностью и поддерживает концепцию исключений. Информацию о различиях между PHP4 и PHP5 вы найдете на сайте *http://www.php.net/manual/en/migration5.php*.

Помимо самого PHP было бы далеко не лишним загрузить и установить *phpMyAdmin*. *phpMyAdmin* – это инструмент администратора баз данных, написанный на языке PHP и упрощающий решение повседневных задач администрирования баз данных MySQL. С его помощью можно решать такие задачи, как создание и удаление баз данных и таблиц, управление привилегиями, ключами и полями и многое, многое другое. Кроме того, он являет собой отличный пример исходных текстов реализации обращения к базам данных MySQL из программного кода PHP! Загрузить *phpMyAdmin* можно с сайта *http://www.phpmyadmin.net*. После установки этого продукта вы сможете открыть в своем броузере URL *http://localhost/phpMyAdmin*, и перед вами появится страница, показанная на рис. 25.3.

Выполнять PHP4 можно не только как модуль, но и как программу CGI, запускаемую веб-сервером, что связано с некоторыми дополнительными издержками. При выполнении PHP в качестве CGI-программы каждая новая страница, содержащая код PHP, требует запуска нового экземпляра интерпретатора PHP, что, в свою очередь, требует создания нового процесса и загрузки в него интерпретатора PHP. Когда интерпретатор заканчивает создание страницы, его процесс завершается, память освобождается, закрываются все дескрипторы файлов и соединения с базами данных.

В качестве модуля веб-сервера интерпретатор PHP становится частью веб-сервера и постоянно загружен в память. Кроме того, он может сохранять такие ресур-

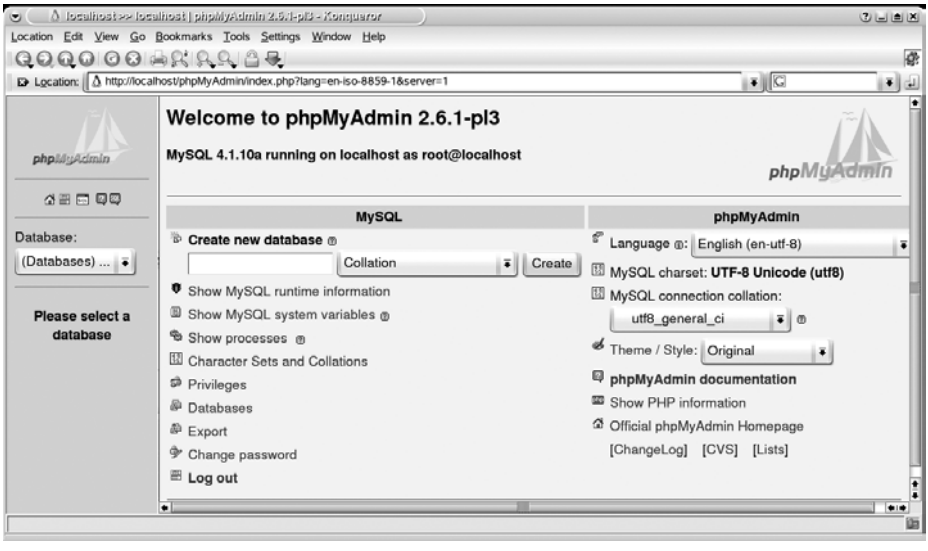


Рис. 25.3. *Администрирование баз данных MySQL с помощью phpMyAdmin*

сы, как соединения с базами данных, при переходе к другим страницам, что значительно повышает производительность.

На всех интенсивно посещаемых сайтах, использующих PHP, он загружен именно в виде модуля, поскольку это обеспечивает более высокую производительность.

PHP4 как модуль Apache

Как мы уже отмечали, для достижения лучшей производительности следует запускать интерпретатор PHP в качестве модуля веб-сервера. В настоящее время большинство дистрибутивов, в том числе Slackware, Debian, SUSE и Red Hat, включают в себя как Apache, так и модуль PHP4 для Apache, поэтому обычно нет необходимости самостоятельно компилировать модуль PHP4. Однако иногда такое решение может оказаться правильным.

Для поддержки обширной функциональности модулю PHP4 требуется большое количество дополнительных библиотек или модулей. Если устанавливать модуль PHP4 с инсталляционного компакт-диска, то программа установки автоматически установит необходимые модули. Однако модули, поставляемые с дистрибутивами, обычно содержат функции, способные удовлетворить все вкусы и потребности. В результате система может оказаться более тяжеловесной и медлительной, чем вызывается необходимостью.

Поэтому преимущество самостоятельной компиляции PHP4 состоит в том, что вы можете решить, какую функциональность включить в этот модуль. Проверьте по документации, какие дополнительные библиотеки вам может потребоваться установить.

Поскольку мы твердо убеждены в необходимости знать о том, что творится за кулисами, даже если используются более удобные готовые решения, то дадим несколько советов относительно того, как начать работу с чистого листа, и как различные части взаимодействуют друг с другом.

Для загрузки модуля PHP4 в веб-сервер Apache во время его работы необходимо присутствие модуля Apache *mod_so*. Проверить его наличие можно с помощью команды:

```
owl$ httpd -l
Compiled-in modules:
        http_core.c
        mod_so.c
```

Если этот модуль недоступен, проверьте, не пропустили ли вы установку каких-либо дополнительных пакетов Apache из вашего дистрибутива. Если вы компилировали Apache самостоятельно, прочтите в документации, как получить этот модуль.

Можно также скомпилировать модуль PHP4 непосредственно в Apache, но это требует тесной связи компиляции Apache и PHP4 и не дает особых преимуществ, поэтому мы не станем освещать здесь эту тему.

Теперь нужно скомпилировать PHP, чтобы сделать из него динамический разделяемый объект (Dynamic Shared Object, DSO). К счастью, это не так сложно, как может показаться. Загрузите PHP4 с *http://www.php.net/download.php*. В итоге вы получите пакет с именем *php-4.4.0.tar.gz* (фактический номер версии может быть несколько иным). Распакуйте *tar*-файл и настройте PHP командой:

```
owl$ ./configure \
                --with-mysql=/usr/lib/mysql\
                --with-ldap=yes\
                --with-gd=yes\
                --with-zlib=yes\
                --with-config-file-path=/etc/\
                --with-apxs=/usr/lib/apache/apxs\
                --enable-versioning\
                --enable-track-vars\
                --enable-thread-safety
```

В пространной документации PHP можно прочесть о многочисленных дополнительных параметрах, но для начала достаточно этих. Учтите, что может потребоваться заменить некоторые из указанных путей теми, которые соответствуют фактическим адресам в вашей системе. После того как команда *configure* завершит свою работу, запустите команду *make*, а затем *make install*, чтобы установить PHP (может потребоваться выполнить *make install* как *root*).

Теперь отредактируйте *httpd.conf*, файл с настройками Apache. Если вы установили Apache с инсталляционного диска, возможно, что следующие строки там уже есть и нужно лишь раскомментировать их. Во всяком случае в *httpd.conf* должны быть следующие строки:

```
LoadModule php4_module libexec/libphp4.so
AddModule mod_php4.c

AddType application/x-httpd-php .php
```

Возможно, вам придется отыскать строку `DirectoryIndex` и изменить ее, чтобы сценарии PHP могли использоваться в качестве страницы по умолчанию:

```
DirectoryIndex index.html index.php
```

Теперь перезапустите Apache:

```
owl# apachectl restart
```

(В некоторых дистрибутивах команда *apachectl* может иметь иное название, попробуйте запустить команду *rcapache*.) После того как сервер будет перезапущен, нужно проверить, правильно ли загружается модуль PHP4. Для этого можно написать небольшую программу на языке PHP, например:

```
<?php
phpinfo();
?>
```

Сохраните этот файл как *phpinfo.php* в каталоге *htdocs* своего установочного каталога Apache (обычно */usr/local/httpd/htdocs*). Теперь вы должны загрузить эту страницу в свой веб-броузер, обратившись по адресу *http://localhost/phpinfo.php*. Если все в порядке, на странице будет показана конфигурация модуля PHP4.

Сервер LAMP в действии

Теперь все компоненты сервера LAMP установлены, и пора запустить несколько примеров.

Если вы еще не сделали этого, читая предыдущий раздел, советуем проверить сейчас свою установку с помощью очень простого файла PHP. Сохраните программный код PHP, который приводился в предыдущем разделе, в файле с именем *info.php*.

Поместите этот файл в каталог, в котором ваш сервер Apache ищет файлы контента. Часто им оказывается */usr/local/httpd/htdocs*, и в нем могут уже находиться файлы, которые ваш дистрибутив поместил туда во время установки (во всяком случае, если вы устанавливали Apache с инсталляционного носителя). Если что-то не так, найдите файл с настройками Apache *httpd.conf*. Часто этот файл находится в каталоге */etc/httpd/*, но если в вашей системе это не так, поищите его командой:

```
locate httpd.conf
```

Найдите в этом файле строку, начинающуюся с `DocumentRoot`. Нужно найти указанный в ней каталог, в котором должен находиться подкаталог *htdocs*, и поместить туда файл *info.php*. Теперь, используя любой броузер, обратитесь по URL *http://localhost/info.php*. В результате вы получите некоторую информацию о настройке вашего модуля PHP.

В PHP есть ряд встроенных функций для работы с данными, хранящимися в MySQL (и других базах данных).

Реляционная база данных состоит из ряда таблиц. Если у вас есть необходимые права доступа, PHP может посылать запросы и обрабатывать данные в этих таблицах. Теперь мы можем написать несколько сценариев PHP для работы с таб-

лицами базы данных. Мы предполагаем, что вы создали базу данных *test_database* и таблицу `comment_table`, а также пользователя *olof*, как было описано выше.

С помощью любого текстового редактора введите следующий код, создающий небольшую HTML-страницу, с помощью которой можно помещать данные в эту таблицу посредством HTML-формы:

```
<html>
<?php
if (isset($_REQUEST["comment"])) {
    $conn = mysql_connect("localhost", "olof", "secret")
      or die("Could not connect to MySQL as olof");

    mysql_select_db("test_database", $conn)
      or die("could not select the test_database");

    if (get_magic_quotes_gpc( )) {
        $comment = stripslashes($_REQUEST["comment"]);
    } else {
        $comment = $_REQUEST["comment"];
    }

    $query = "INSERT INTO comment_table VALUES ('0', '"
      . mysql_real_escape_string($comment) . "')";

    mysql_query($query)
      or die(mysql_error( ));
}
?>

<form action="" method="POST">
    <input type="text" name="comment" size="80"><br>
    <input type="submit">
</form>

</html>
```

При работе с любой базой данных необходимо соблюдать меры предосторожности, чтобы исключить возможность манипулирования SQL-запросами в результате некорректной обработки ввода, получаемого от пользователя. Если этого не сделать, злоумышленник сможет просто украсть базу данных. Вы можете обезопасить себя, если будете выполнять преобразование данных перед вставкой их в тело запроса SQL. Обычно вполне достаточно пропустить данные, введенные пользователем, через функцию `mysql_real_escape_string()`. В некоторых ситуациях, возможно, потребуется сначала обработать данные функцией `stripslashes()`. Это приходится делать из-за специализированной возможности PHP под названием `magic_quotes_gpc`, которая предназначалась для автоматического обеспечения безопасности входных данных при работе с базами данных. Сама идея вполне достойная, но ее реализация не обеспечивает достаточную степень безопасности и создает некоторые проблемы для программистов. Мы рекомендуем отключить эту особенность в своих настройках. Иначе вам сначала придется узнать, активирована ли она, и затем нейтрализовать возникающие побочные эффекты, если таковые будут обнаружены.

Чтобы выполнить этот сценарий, сохраните его как файл с расширением *.php*, скопируйте в каталог документов вашего веб-сервера и обратитесь к сценарию

через веб-броузер. Например, если вы сохранили сценарий как файл *edit.php*, можно перейти по URL *http://localhost/edit.php*, и сценарий будет выполнен. Веб-сервер знает, что все, находящееся между тегами <?php и ?>, должно быть пропущено через модуль PHP. Таким образом, код PHP можно непосредственно встраивать в страницу HTML.

Теперь, когда мы можем вводить комментарии в нашу базу данных, мы хотим посмотреть на них. Поэтому приведем пример сценария, читающего информацию из базы данных:

```
<html>
<?php
$conn = mysql_connect("localhost", "olof", "secret")
  or die("Could not connect to MySQL as olof");

mysql_select_db("test_database", $conn)
  or die("could not select the test_database");

$query = "SELECT * FROM comment_table";
$result = mysql_query($query)
  or die(mysql_error( ));

$numbers_cols = mysql_num_fields($result);

print "<b>query: $query</b>";
print "<table border=1>\n";
print "<tr>";
print "<td>ID</td>";
print "<td>Comment</td>";
print "</tr>";

while (list($id, $comment) = mysql_fetch_array($result)) {
    print "<tr>";
    print "<td>" . htmlspecialchars($id, ENT_QUOTES) . "</td>";
    print "<td>" . htmlspecialchars($comment, ENT_QUOTES) . "</td>";
    print "</tr>";
}

print "</table>";

?>
</html>
```

Как видите, мы использовали теги HTML для разметки таблиц, в которых показывается содержимое базы данных, что вполне естественно и очевидно. Кроме того, обратите внимание, что данные, полученные из базы, не выводились на страницу без предварительной обработки. Это позволило бы злоумышленнику разрушить представление страницы некорректно введенными данными. По этой причине все данные выводились с использованием функции `htmlspecialchars()`.

Мы стремились привести по возможности более простые примеры, чтобы не перегружать вас большим объемом информации. Если вы хотите глубже окунуться в удивительный мир LAMP, советуем прочесть хорошую книгу «Web Database Applications with PHP & MySQL» (O'Reilly) или «MySQL/PHP Database Applications» (John Wiley & Sons).

26

Система безопасности

В этой главе мы обсудим основы системы безопасности Linux. К сожалению, тема безопасности приобретает все большее значение, особенно с ростом постоянно подключенных к сети систем, уязвимых для удаленных атак даже тогда, когда на них не работают.

По большей части безопасность является просто практическим проявлением здравого смысла. Многие хорошие приемы являются также и простейшей, но часто игнорируемой практикой, о чем мы и расскажем в первую очередь. Затем мы перейдем к некоторым менее очевидным правилам и в завершение обсудим сложную тему сетевой безопасности. Мы также приведем ряд способов использования сетевых экранов (брандмауэров), с помощью которых простые конфигурации могут быть защищены от сетевых атак.

Общий взгляд на систему безопасности

Не всегда легко получить взвешенное представление о проблемах безопасности компьютерных систем. Средства массовой информации склонны создавать сенсации из историй со взломами системы защиты, особенно если они касаются известных компаний или учреждений. С другой стороны, управление системой безопасности часто оказывается технически сложной и трудоемкой задачей. Многие пользователи Интернета придерживаются точки зрения, что их машины не хранят ценных данных, а потому обеспечение безопасности не является для них проблемой. Другие тратят огромные силы на то, чтобы укрепить защиту своих систем от несанкционированного использования. Независимо от того, к какой крайней позиции в этом спектре вы ближе, следует иметь в виду, что для вашей системы защиты всегда существует риск стать объектом атаки. Есть тьма причин, по которым злоумышленники могут заинтересоваться взломом защиты вашей системы. Ценность имеющихся в системе данных – лишь одна из них. Далее в этой главе мы обсудим и другие причины. Вы сами должны решить, какие затраты можете позволить себе на обеспечение безопасности, но, по нашему мнению, лучше перестраховаться.

Традиционно предметом забот была безопасность систем, доступ к которым осуществлялся с подключенных проводами терминалов или системной консоли.

В этой сфере наибольшая опасность обычно исходила изнутри самой организации – владельца системы, и лучшим видом защиты была физическая безопасность, то есть размещение системных консолей, терминалов и машин в закрытых для доступа помещениях. Даже когда компьютерные системы стали объединяться в сети, доступ к ним оставался очень ограниченным. Получение доступа к имевшимся сетям обходилось очень дорого либо это были закрытые сети, не допускавшие подключений из произвольной точки.

Распространение Интернета послужило толчком к возникновению проблем сетевой безопасности. Компьютер, подключенный к Интернету, потенциально подвержен угрозе со стороны десятков миллионов машин, разбросанных по всему свету. С облегчением доступа к сети растет и число антиобщественно настроенных людей, сосредоточенных на том, чтобы чинить другим неприятности. Для системного администратора представляет интерес ряд видов асоциального поведения в Интернете. В этой главе нас будут интересовать следующие из них:

Отказ в обслуживании (Denial of service, или DoS)

Этот вид атаки снижает эффективность сервиса, предоставляемого системой, или вообще срывает его работу.

Незаконное проникновение (Intrusion)

При этом виде атаки осуществляется доступ к системе путем подбора паролей или компрометации некоторого сервиса. Получив доступ к системе, нарушитель может испортить или украсть данные либо воспользоваться системой для организации атак на другие машины.

Подслушивание (Snooping)

Этот вид атаки состоит в перехвате данных других пользователей, паролей или другой конфиденциальной информации. Иногда при такой атаке осуществляется и подмена данных. Обычно это делается путем перехвата данных, передаваемых в сети, а может и путем компрометации системы с целью перехвата библиотечных или системных вызовов, несущих секретную информацию (например, пароли).

Вирусы, черви и троянские кони (viruses, worms и Trojan Horses)

Любая такая атака состоит в том, чтобы заставить пользователя выполнить программу, созданную атакующим. Эти программы можно получить по электронной почте, с веб-сайта или даже внутри некоторой другой, по виду безвредной программы, загруженной из Интернета и установленной локально.

Атака типа DoS обычно заключается в генерировании огромного количества запросов к сервису, предоставляемому системой. Такая повышенная активность может привести к нехватке системной памяти, вычислительной мощности или пропускной способности сети. В результате система отказывается выполнять последующие запросы либо скорость ее работы становится ниже допустимой. Для действенности данной атаки злоумышленник должен воспользоваться ошибками в проектировании сервиса или суметь сгенерировать запросы в количестве, значительно превышающем мощность сервиса.

Более коварной формой DoS-атаки является распределенная атака по типу отказа в обслуживании (DDoS). В этом варианте генерация запросов к сервису организуется с большого числа машин. Тем самым повышается разрушительное дей-

ствие DoS-атаки по двум направлениям: цель оказывается поражена огромным объемом трафика, а злоумышленник может скрыться среди тысяч ничего не подозревающих участников атаки. Использование большого числа машин, с которых ведется атака, особенно осложняет взятие ведущихся DDoS-атак под контроль и исправление последствий. Даже те, кого не заботит состояние собственных данных, должны принимать меры защиты против таких атак, чтобы снизить риск стать невольным соучастником какой-либо DDoS-атаки.

Второй вид атаки, который иногда называют взломом, или *крекингом* (*cracking*), чаще всего ассоциируется в сознании людей с безопасностью.[1] Многие компании и учреждения хранят секретные данные в компьютерных системах, доступных через сеть. Обычный предмет беспокойства среднего пользователя Интернета – это хранение данных кредитных карточек веб-сайтами. Когда дело касается денег, у нечестных людей возникает побуждение получить доступ к таким конфиденциальным данным и украсть их или незаконно использовать.

Иногда для получения неавторизованного доступа или срыва работы сервисов используются очень изобретательные методы, хотя и не являющиеся этичными. Для разработки алгоритма вторжения нередко требуется хорошее знание взламываемой системы, позволяющее использовать ее слабость. Часто обнаруженный способ вторжения оформляется в виде так называемого *rootkit* – набора программ или сценариев, которыми каждый, кто обладает лишь элементарными знаниями, может воспользоваться для проникновения через пробел в системе защиты. Подавляющее большинство атак незаконного проникновения осуществляется лицами, которых называют «script kiddies», пользующимися такими готовыми инструментальными наборами, не имея никакого представления о системах, которые они атакуют. К счастью, системному администратору обычно легко защитить систему от таких хорошо известных атак. В этой главе мы обсудим различные способы защиты своей системы.

Первые шаги в организации защищенной системы

Есть ряд элементарных вещей, которые можно сделать, чтобы защитить Linux-систему от простейших угроз безопасности. Конечно, в зависимости от конкретной конфигурации системы, способов ее применения и т. д., эти действия могут оказаться сложнее, чем описываемые здесь простые настройки. В этом разделе мы кратко расскажем об основных механизмах защиты Linux от большинства стандартных атак – это базовый подход, используемый одним из авторов при установке новой системы.

---

[1] Термины «cracking» и «hacking» (*крекинг* и *хакинг*) часто путают в обиходе. В то время как *cracking* осуществляет аморальное или незаконное поведение (такое как компрометация системы защиты), *hacking* – общий термин, относящийся к программированию, ковырянию в чем-то, проявлению большого интереса. В популярных изданиях часто «хакерством» называют то, что в действительности является «крекерством». Сообщество Linux стремится восстановить положительный подтекст слова «хакер».

Выключение ненужных сетевых демонов

Первым шагом в защите Linux-машины является останов или отключение всех сетевых демонов, которые вам не требуются. В целом любой сетевой порт, на котором система ожидает соединений, порождает риск, поскольку может существовать *эксплойт* (способ использования слабости в системе защиты) против демона, использующего этот порт. Быстро узнать, какие порты открыты, можно с помощью *netstat -an*, как показано ниже (некоторые строки мы укоротили):

```
# netstat -an
Active Internet connections (servers and established)
Proto Recv-Q Send-Q Local Address           Foreign Address     State
tcp        0      0 0.0.0.0:7120            0.0.0.0:*           LISTEN
tcp        0      0 0.0.0.0:6000            0.0.0.0:*           LISTEN
tcp        0      0 0.0.0.0:22              0.0.0.0:*           LISTEN
```

Мы видим, что система ожидает соединений на портах 7120, 6000 и 22. Из файла */etc/services* или с помощью параметра *-p* команды *netstat* часто можно узнать, какие демоны связаны с этими портами. В данном случае это сервер шрифтов X, сервер X Window System и демон *ssh*.

Если вы обнаружите много других открытых портов для таких демонов, как *telnetd*, *sendmail* и т. д., спросите себя, действительно ли вам нужно, чтобы работали эти демоны. Время от времени объявляется о появлении эксплойтов для различных демонов, и если вам не удается старательно следить за этими обновлениями, касающимися безопасности, ваша система может оказаться уязвимой для атаки. Кроме того, *telnetd*, *ftpd* и *rshd* осуществляют передачу паролей для аутентификации открытым текстом через Интернет. Гораздо лучше применять *sshd*, который шифрует передаваемые данные и использует более сильный алгоритм аутентификации. Даже если вы не применяете *telnetd*, оставлять его работающим на вашей машине не очень разумно, поскольку не исключено, что кто-нибудь найдет способ проникновения в систему с его помощью.

Для останова сервисов обычно достаточно отредактировать соответствующие файлы с настройками, зависящие от вашего дистрибутива, и перезагрузить систему (чтобы гарантировать, что их не стало). В системах Red Hat, например, многие демоны запускаются сценариями, расположенными в каталоге */etc/rc.d/init.d*. Помешать запуску соответствующих демонов можно, переименовав или удалив эти сценарии. Другие демоны запускаются демонами *inetd* или *xinetd* в ответ на входящие сетевые соединения. Модифицировав их[1] файлы с настройками, можно ограничить количество демонов, выполняющихся системой.

Если работа некоторого сервиса на вашей машине абсолютно необходима (например, X-сервера), постарайтесь запретить соединения с этим сервисом для непрошенных хостов. Например, надежнее всего разрешить соединения *ssh* только определенным доверенным хостам, таким как машины из вашей локальной сети. В случае сервера X и сервера шрифтов X, которые работают на многих настольных машинах Linux, обычно вообще нет оснований разрешать соединения с ними кому-либо, кроме самого этого локального хоста. Ограничение соединений с этими демонами можно выполнить с помощью TCP-оболочек или IP-фильтрации, которые будут описаны в этой главе.

---

[1] Имеются в виду настройки `inetd` и `xinetd`. – *Примеч. науч. ред.*

«Горячая десятка» ошибок, которые делать нельзя

Мы сделали заявление, что обеспечение безопасности – это в основном проявление здравого смысла, так в чем он заключается в данном случае? В этом разделе мы приводим наиболее частые ошибки в организации защиты. (В действительности в этом списке нет десяти ошибок, но их достаточно, чтобы воспользоваться стандартной фразой «горячая десятка».) Систематически избегать их несколько труднее, чем может показаться вначале.

Не используйте простые или легко угадываемые пароли

Никогда не используйте пароль, который совпадает или похож на ваш ID пользователя, имя, дату рождения, название компании или кличку вашей собаки. Если вы радиолюбитель, не используйте свой позывной; если вы любите машины, не используйте название фирмы, модели или регистрационный номер своей машины – идея должна быть понятна. Ваши пароли не должны быть простыми словами, которые можно найти в словаре. Лучшие пароли – бессмысленные строки. Есть одно хорошее правило: использовать пароль, основанный на простом правиле и фразе, которую вы можете запомнить. Например, можно выбрать такое правило: выбрать последнюю букву каждого слова во фразе «Mary had a little lamb, its fleece was white as snow». Получится пароль `ydaebsesesw`, который не так легко угадать, но легко запомнить. Другой распространенный прием – использование в пароле цифр и знаков пунктуации; на практике некоторые программы *passwd* этого требуют. Еще лучше комбинировать оба эти приема. Один из моих коллег использует команду *head -c6 /dev/random | mimencode* для генерации случайных паролей. Регулируя число генерируемых случайных байтов (*-c6*), можно управлять длиной генерируемых паролей. Так, задавая на входе шесть символов, на выходе мы имеем восемь символов – максимальная длина пароля в некоторых дистрибутивах Linux.

Не пользуйтесь учетной записью root без необходимости

Одна из причин, по которым многие распространенные настольные операционные системы, например Windows, столь уязвимы для атак вирусов, распространяемых через электронную почту, и им подобных, заключается в отсутствии всесторонней системы прав доступа. В таких системах любому пользователю разрешены доступ к любому файлу, выполнение любой программы и перенастройка системы любым образом. Из-за этого легко вынудить пользователя выполнить программу, которая может нанести системе реальный ущерб. Напротив, модель безопасности Linux предоставляет право выполнения большого числа привилегированных задач, таких как установка нового программного обеспечения или модификация файлов конфигурации, только одному пользователю – *root*. Не поддавайтесь соблазну всегда использовать учетную запись *root*! В таком случае вы отказываетесь от одного из самых мощных средств защиты против вирусов и троянских коней (не говоря уже о случайных командах *rm -rf* *!). Всегда регистрируйтесь как обычный пользователь и применяйте команды *su* или *sudo*, чтобы временно получить права *root* при необходимости выполнить привилегированную задачу. Есть и еще одно дополнительное преимущество в таком ограниченном использовании учетной записи *root*: регистрация в журнале. Команды *su* и *sudo* при своем вызове помещают в файл системного журнала записи с указанием ID пользовате-

ля, выполнившего *su* или *sudo*, а также дату и время выполнения этой команды. Это очень помогает проследить, кто и когда пользовался правами *root*.

Не раскрывайте свои пароли

Никогда никому не сообщайте свой пароль. Это также означает, что вы не должны записывать свой пароль на стикерах, приклеиваемых к монитору, или в тетради, хранящейся в верхнем ящике стола. Если вам нужно предоставить кому-либо временный доступ к своей системе, создайте для него отдельную учетную запись. Это даст вам некоторую возможность проследить за тем, что делали на вашей машине, и затем легко убрать последствия. Если вам действительно необходимо разрешить кому-то войти на вашу машину как *root*, воспользуйтесь командой *sudo*, позволяющей предоставлять пользователям права *root* без раскрытия пароля суперпользователя.

Не доверяйте безоглядно загружаемым исполняемым модулям

Очень удобно устанавливать на машине программы, получаемые в двоичном виде, но прежде чем запускать исполняемые модули, следует задуматься о том, насколько вы им доверяете. Если вы устанавливаете программные пакеты, загруженные прямо с официального сайта вашего дистрибутива или серьезного сайта разработчиков, то можно быть достаточно уверенным в безопасности программ. Если вы получаете их с неофициального зеркального сайта, следует подумать о том, насколько вы доверяете его администраторам. Не исключено, что кто-то распространяет модифицированное программное обеспечение с «черным входом», который позволит получить несанкционированный доступ к вашей машине. Такой подход несколько параноидален, но он принят многими организациями, распространяющими Linux. Например, Debian разрабатывает средства для проверки отсутствия модификации программных пакетов. Для других дистрибутивов готовятся аналогичные меры защиты целостности пакетов программного обеспечения.

Если вы хотите установить и выполнить программу, которую получили в двоичном виде, то должны принять некоторые меры, снижающие риск. К сожалению, этими приемами непросто воспользоваться новичкам в среде Linux. Во-первых, никогда не выполняйте программы, которым не вполне доверяете, с привилегиями *root*, если только права *root* не требуются программе специально. В результате ущерб, возможно, наносимый программой, будет ограничен только файлами и каталогами конкретного пользователя. Во-вторых, если вы хотите получить некоторое представление о том, что делает программа, не выполняя ее, можно применить к исполняемым модулям команду *strings*. Она покажет все жестко закодированные строки, которые есть в программном коде. Следует поискать в них обращения к важным файлам или каталогам, таким как */etc/passwd*, */bin/login* и т. д. Обнаружив ссылку на важный файл, следует подумать о том, согласуется ли она с назначением рассматриваемой программы. Если нет, будьте осторожны. При достаточной технической подготовке можно сначала запустить программу и посмотреть, что она делает, с помощью таких программ, как *strace* или *ltrace*, которые показывают системные и библиотечные вызовы, осуществляемые программой. Поищите в трассировке следы необычных операций с файловой системой или сетью.

Не упускайте из виду файлы журналов

Файлы системных журналов – ваши друзья, которые могут многое рассказать о происходящем в системе. Вы можете найти в них сведения о том, когда происходили сетевые соединения с вашей системой, кто пользовался учетной записью *root*, были ли неудачные попытки входа в систему. Нужно периодически проверять файлы журналов и научиться определять, где нормальная работа, а где, что еще важнее, ненормальная. Обнаружив что-то необычное, проведите расследование.

Не допускайте слишком сильного устаревания системы

Необходимо поддерживать программное обеспечение на машине в достаточно обновленном состоянии. Та машина Linux с ядром версии 1.2, которая стоит у вас в углу и годами отлично обслуживает принтеры, послужит хорошей темой для разговора на вечеринке, но в любой момент может стать причиной инцидента в системе защиты. Своевременное обновление программного обеспечения гарантирует, что в нем будут устранены все выявленные программные ошибки и бреши в системе безопасности. Большинство дистрибутивов Linux предоставляют ряд пакетов, являющихся исключительно исправлениями в системе защиты, поэтому поддержание защищенности системы не связано с такими проблемами, как изменение файлов с настройками и функций. Необходимо, по крайней мере, следить за появлением таких обновлений.

Не забывайте о физической безопасности

Проникновение через защиту компьютерных систем чаще всего осуществляется сотрудниками тех организаций, в которых они установлены. Самая тщательная настройка программной защиты не поможет, если кто-то может подойти к машине и запустить с дискеты код эксплойта. Если в вашей машине установлены BIOS или системное PROM, позволяющие изменять порядок устройств для начальной загрузки, сделайте так, чтобы загрузка с дискет и CD-ROM происходила после неудачной попытки загрузки с жесткого диска. Если настройку BIOS можно защитить паролем, воспользуйтесь этой функцией. Если можно повесить замок на корпус машины, сделайте это. Если можно поставить машину в физически защищенном месте, например в запертом помещении, еще лучше.

Настройка TCP-обертки

Ранее мы говорили о том, что подключение машины к сети существенно повышает угрозу атаки на нее. Разобравшись с тем, что в отношении защиты советует здравый смысл, можно более пристально взглянуть на проблемы сетевой безопасности. Сейчас мы опишем простой, но действенный метод уменьшения риска нежелательного доступа к системе через сеть с помощью средства, называемого *TCP-оберткой* (TCP wrapper). Этот механизм создает «обертку», в которую помещается имеющийся сервис, например почтовый сервер, чтобы осуществлять фильтрацию подключений к нему из сети, отказывая в соединении неавторизованным сайтам. Это простой способ добавить управление доступом к сервисам, которые разрабатывались без его учета. Чаще всего этот метод применяется вместе с демонами *inetd* или *xinetd*.

TCP-обертка – некий эквивалент охранников, или «вышибал», которых можно встретить у входа на крупные собрания людей или в ночные клубы. Когда вы приходите в такое место, то первым делом встречаете охранника, который может поинтересоваться вашим именем и адресом. Затем он сверяется со списком приглашенных, и если находит вас в нем, то отодвигается и пропускает вас внутрь.

Когда осуществляется соединение с сервисом, защищенным TCP-оберткой, вы прежде всего сталкиваетесь с этой оберткой. Она интересуется именем хоста или адресом, откуда исходит соединение, и проверяет его по списку, ограничивающему доступ. Если источник указан в списке, обертка разрешает сетевому соединению доступ к фактической программе-демону.

Использовать TCP-обертки можно двумя способами в зависимости от дистрибутива Linux и конфигурации. Если управление сервисами осуществляется демоном *inetd* (проверьте, существует ли файл */etc/inetd.conf*), то TCP-обертки реализуются с помощью специального демона под названием *tcpd*. Если же используется демон *xinetd* (проверьте, существует ли каталог */etc/xinetd.d*), то обычно настраивают *xinetd* для непосредственного использования TCP-обертки. Оба случая будут рассмотрены в последующих разделах.

Использование TCP-оберток с inetd

Если ваша система запускает сетевые сервисы с помощью демона *inetd*, то для использования TCP-оберток может оказаться необходимым отредактировать файл */etc/inetd.conf*. Возьмем в качестве примера демон сервиса finger, *in.fingerd*. Идея состоит в том, чтобы вместо демона *in.fingerd* посредством *inetd* запускался демон *tcpd*. Демон *tcpd* выполняет операцию TCP-обертки, а затем запускает вместо нее *in.fingerd*, если соединение разрешено.

Настройка для TCP-оберток требует очень простой модификации */etc/inetd.conf*. Для демона finger в этом файле первоначально может существовать такая запись:

```
# /etc/in.fingerd демон finger
finger    stream tcp nowait root /usr/sbin/in.fingerd in.fingerd
```

Чтобы защитить демон finger с помощью *tcpd*, модифицируйте запись в */etc/inetd.conf* следующим образом:

```
# /etc/in.fingerd демон finger
finger    stream tcp nowait root /usr/sbin/tcpd /usr/sbin/in.fingerd
```

Мы организовали выполнение команды *tcpd* вместо фактической команды *in.fingerd*. Полный путь к демону finger передается *tcpd* в качестве *аргумента*, и *tcpd* использует этот аргумент для запуска реального демона после проверки, разрешен ли доступ.

Такую модификацию нужно осуществить для каждой программы-демона, которую вы хотите защитить. Большинство систем Linux уже настроены на запуск *tcpd*, поэтому необходимость в этих изменениях может отсутствовать.

Использование TCP-оберток с xinetd

xinetd служит заменой *inetd*, принятой в некоторых дистрибутивах (например, Red Hat). В большинстве случаев *xinetd* содержит встроенную поддержку TCP-

оберток, поэтому вам потребуется только модифицировать файлы с настройками TCP-обертки (*/etc/hosts.allow* и */etc/hosts.deny*), что будет описано в следующем разделе. Если вы устанавливаете *xinetd* сами, необходимо скомпилировать поддержку TCP-оберток, как описано в документации по *xinetd*.

Файлы /etc/hosts.allow и /etc/hosts.deny

TCP-обертки используют два файла с настройками – */etc/hosts.allow* и */etc/hosts.deny* – для определения правил доступа к каждому сетевому демону, защищенному TCP-обертками. Эти файлы подробно описаны на странице справочного руководства *hosts_access*, но мы покажем здесь их механизм, поскольку в стандартном случае они довольно просты.

Когда вызывается TCP-обертка, она получает IP-адрес подключающегося хоста и пытается определить его имя с помощью обратного поиска DNS. Затем она справляется в файле */etc/hosts.allow*, разрешен ли именно этому хосту доступ к запрашиваемому сервису. Если хост найден в списке, ему разрешается доступ и вызывается фактический сетевой демон. Если соответствия в файле */etc/hosts.allow* не найдено, по файлу */etc/hosts.deny* проверяется, запрещен ли доступ конкретно этому хосту. Если найдено соответствие в этом файле, соединение закрывается. Если хост не найден ни в том, ни в другом файле, то доступ разрешается. Эта простая технология достаточно мощна, чтобы удовлетворить большинству требований ограничения доступа.

Синтаксис *hosts.allow* и *hosts.deny* довольно прост. Каждый файл содержит набор правил. Каждое правило обычно занимает одну строку, но может быть разбито на несколько строк с помощью символа обратного слэша в конце строки. Общий вид правила следующий:

```
daemon_list : client_list : shell_command
```

`daemon_list` – это список демонов, разделенных запятыми, к которым относится данное правило. Демоны указываются с помощью основного имени их команды, то есть фактически выполняемой программы демона, осуществляющей обработку запрошенного сервиса. `client_list` – это список имен хостов или IP-адресов, разделенных запятыми, к которым применяется правило. Мы продемонстрируем это ниже на примере. Параметр `shell_command` является необязательным и определяет команду, которая будет выполнена, если правило применимо. С помощью такой команды можно, например, регистрировать в журнале входящие соединения.

daemon_list и *client_list* могут содержать шаблоны, соответствующие группе демонов или хостов, что позволяет не перечислять каждый из них отдельно. Кроме того, есть ряд лексем, упрощающих чтение и составление правил. Правила составления шаблонов довольно сложны, поэтому мы не станем их подробно обсуждать, а отошлем читателя к странице руководства *hosts_access*.

Начнем с простого файла *hosts.deny*, который выглядит так:

```
# /etc/hosts.deny
ALL: ALL
```

Первая строка – это комментарий. Следующая строка является правилом, которое интерпретируется так: «запретить доступ ко всем (ALL) сервисам со всех (ALL) хостов». Если при этом файл */etc/hosts.allow* пуст, результатом такого пра-

вила будет запрещение доступа к чему бы то ни было со всех машин в Интернете, включая локальный хост! Чтобы решить эту проблему, можно внести в файл простое изменение:

```
# /etc/hosts.deny
ALL: ALL EXCEPT localhost
```

Таким правилом можно надежно пользоваться почти во всех случаях: это обеспечивающее безопасность значение по умолчанию. Напомним, что к правилам */etc/hosts.allow* система обращается раньше, чем к правилам */etc/hosts.deny*, поэтому, добавляя правила в *hosts.allow*, можно отменить это значение по умолчанию в *hosts.deny*. Допустим, например, что мы хотим предоставить любой системе в Интернете доступ к демону сервиса finger. Для этого мы добавим в */etc/hosts.allow* правило, которое выглядит так:

```
# /etc/hosts.allow
in.fingerd: ALL
```

Чаще всего TCP-обертки применяют для того, чтобы ограничить группу хостов, которым разрешен доступ к некоторому сервису. Хосты могут указываться с помощью IP-адреса, имени хоста или некоторого шаблона адреса или имени (например, чтобы задать группу хостов). Допустим, что мы хотим сделать демон finger доступным только небольшой группе доверенных хостов. В таком случае модифицируем наш файл *hosts.allow* следующим образом:

```
# /etc/hosts.allow
in.fingerd: spaghetti.vpasta.com, .vpizza.com, 192.168.1.
```

В этом запросе мы разрешили FTP-запросы с хоста с именем `spaghetti.vpasta.com`, а также со всех хостов в домене `vpizza.com` и с любой системы, IP-адрес которой начинается с `192.168.1`.[1]

Важно разобраться с правилами соответствия хостов и IP-адресов в *hosts.allow* и *hosts.deny*, в которых присутствие и положение точки является критическим. Если шаблон начинается с точки, предполагается, что это имя домена, к которому должна принадлежать система, посылающая запрос. Если шаблон оканчивается точкой, предполагается, что это шаблон IP-адреса. Есть и другие способы задания групп хостов, в том числе с помощью сетевых групп NIS и явных масок IP-адресов. Полное описание синтаксиса этих шаблонов есть на странице справочного руководства *hosts_access*.

Брандмауэры: фильтрация IP-пакетов

TCP-обертки можно использовать для того, чтобы разрешить доступ к определенным сервисам на машине определенной группе хостов, но часто желательно организовать более тонкий контроль над пакетами, поступающими в данную систему (или посылаемыми из нее). Нужно также учесть, что TCP-обертки действуют только в отношении сервисов, настроенных на работу с *inetd* или *xinetd*, тогда как некоторые сервисы (например, *sshd* в некоторых системах) являются «автономными» и обладают собственными функциями управления доступом. Есть и другие сервисы, в которых не реализовано никакого контроля доступа,

[1] А это могут быть только хосты локальной сети. – *Примеч. науч. ред.*

поэтому необходим другой уровень защиты, если мы хотим управлять соединениями с этими сервисами.

В настоящее время для пользователей Интернета стало обычным делом защищать себя от сетевых атак с помощью технологии, называемой IP-фильтрацией. При IP-фильтрации ядро изучает каждый передаваемый или принимаемый пакет и, прежде чем пропустить его, решает, разрешить ли его прохождение, отбросить или каким-то образом модифицировать. IP-фильтрацию часто называют *брандмауэром* (*firewalling*), поскольку путем тщательной фильтрации пакетов, приходящих в систему или покидающих ее, создается некая «стена» между машиной и остальной частью Интернета. IP-фильтрация не защитит вас от вирусов или троянских коней, как и от дефектов в программах, но она создает защиту против многих видов сетевых атак, например некоторых видов DoS-атак и фальсификации IP-адресов (таких как «спуфинг» – пометка пакетов как пришедших не от той системы, которая их в действительности отправила). IP-фильтрация также предоставляет дополнительный уровень контроля, предотвращающий доступ нежелательных пользователей к системе.

Чтобы заставить систему IP-фильтрации действовать, нужно знать, каким пакетам разрешать прохождение, а какие отвергать. Обычно решение о фильтрации пакета принимается на основании его заголовка, в котором содержатся такие данные, как IP-адреса отправителя и получателя, тип протокола (TCP, UDP и т. д.) и номера портов отправителя и получателя (указывающие конкретный сервис, для которого предназначен пакет). Различные сетевые сервисы используют различные протоколы и номера портов. Например, большинство веб-серверов принимают запросы TCP на порт 80. Если мы хотим отфильтровать весь входящий трафик HTTP от нашей системы, нужно организовать IP-фильтр, который отклоняет все пакеты TCP, адресованные порту 80.

Для решения некоторых задач фильтрации недостаточно изучить только заголовок пакета, а требуется проанализировать фактические данные, которые несет пакет. Эту технологию иногда называют проверкой пакетов с учетом состояния протокола (stateful inspection), поскольку пакет рассматривается не изолированно, а в контексте текущего сетевого соединения. Пусть, например, нужно разрешить пользователям нашей локальной сети работать с FTP-серверами, находящимися за ее пределами. FTP – сложный протокол, который использует одно соединение TCP для отправки серверу команд, а другое – для передачи фактических данных. К сожалению, в спецификации FTP не определен конкретный номер порта для передачи данных, поэтому клиент и сервер должны договориться о номерах портов с помощью ряда команд. Если не пользоваться технологией stateful inspection, то для осуществления передачи данных по FTP потребуется разрешить подключения TCP к произвольным портам. Stateful inspection решает эту проблему путем анализа переговоров между клиентом и сервером относительно номеров портов и разрешения прохождения пакетов TCP через выбранный ими порт.

IP-фильтры реализованы в ядре Linux, в котором есть программный код, проверяющий каждый принимаемый и передаваемый пакет и применяющий правила фильтрации, которые решают судьбу этого пакета. Правила настраиваются с помощью утилиты, которая в командной строке принимает от пользователя аргументы и преобразовывает их в спецификации фильтра, сохраняемые ядром и используемые им в качестве правил.

Включенная в ядро Linux IP-фильтрация прошла в своем развитии три поколения, у каждого из которых свой механизм настройки. Первое поколение носило название *ipfw* (от «IP firewall») и обеспечивало основные возможности фильтрации, но было недостаточно гибким и эффективным в сложных конфигурациях. Сейчас *ipfw* используется редко. Второе поколение IP-фильтров, под названием *IP-цепочек* (*IP chains*), было значительно эффективнее *ipfw* и до сих пор широко используется. Последнее поколение фильтров называется *netfilter*/*iptables*. *netfilter* – это компонент ядра, а *iptables* – утилита настройки, выполняемая в пространстве пользователя. Оба названия часто используют взаимозаменяемым образом. *netfilter* не только обладает большей гибкостью в настройке, но и допускает расширение. В следующих разделах мы опишем *netfilter* и приведем примеры некоторых простых конфигураций.

Основы netfilter

netfilter входит в ядро Linux версий 2.4.0 и выше. Основная утилита для настройки и отображения таблиц фильтрации называется *iptables* и включена во все современные дистрибутивы Linux. Команда *iptables* дает возможность настроить большой и сложный набор правил фильтрации пакетов, и потому у нее много параметров командной строки. Мы расскажем здесь о наиболее общеупотребимых. Страница справочного руководства *iptables* содержит полное их описание.

Для начала, чтобы возбудить у вас аппетит, заглянем вперед и покажем, к чему мы должны прийти:

```
iptables -A INPUT -m state --state NEW -m tcp -p tcp --dport 22 -j ACCEPT
```

Эта команда определяет правило фильтрации IP-пакетов для приема новых входящих соединений TCP на порт 22 (служба *ssh*) локальной системы. Она также использует модуль расширения *state*, реализующий механизм отслеживания состояния соединений. Ниже будет рассказано, как все это работает.

Важным понятием *netfilter* является *цепочка* (*chain*), состоящая из списка правил, применяемых к пакетам, когда они приходят в систему, исходят из нее или проходят насквозь. По умолчанию в ядре определены три цепочки, но администратор может создавать новые цепочки и присоединять их к уже имеющимся. Ниже приводится описание трех предопределенных цепочек:

`INPUT`

Эта цепочка применяется к полученным пакетам, адресованным локальной системе.

`OUTPUT`

Эта цепочка применяется к пакетам, передаваемым локальной системой.

`FORWARD`

Эта цепочка применяется, когда пакет должен быть маршрутизирован данной системой с одного сетевого интерфейса на другой. Она используется, когда система выступает как маршрутизатор пакетов или шлюз и не является ни источником, ни получателем пакета.

Каждое правило в цепочке связано с группой критериев, определяющих, к каким пакетам применяется правило и какие действия должны быть совершены над па-

кетами, подпадающими под действие этого правила. Над пакетами могут совершаться такие действия, как принятие пакета (разрешение на прием или передачу), отбрасывание пакета (отказ принять или передать его) или передача пакета в другую цепочку. (Последнее удобно при создании пользовательских цепочек, допускающих иерархическое построение сложных правил фильтрации пакетов.) Пакет проходит через все правила цепочки, пока не будет принят, отброшен или не достигнет конца цепочки; если он достигает конца, к нему применяется операция, установленная для цепочки по умолчанию. В качестве действия цепочки по умолчанию можно назначить принятие или отбрасывание всех пакетов.

В Linux *netfilter* поддерживает ряд других интересных операций, допустимых в правилах фильтрации. Одним из важных достоинств *netfilter* является возможность расширения. Можно разрабатывать расширения, усиливающие функции *netfilter*. Вот некоторые примеры более сложной обработки пакетов:

Регистрация пакетов

Можно создать правила, лишь регистрирующие описание пакета, соответствующего правилу, которое впоследствии будет подвергнуто анализу. Это очень удобно для обнаружения атак и тестирования настроек фильтрации.

Stateful inspection

В *netfilter* есть ряд вспомогательных модулей, поддерживающих технологию stateful inspection, например для управления соединениями FTP, как описано выше.

Преобразование сетевых адресов

Преобразование сетевых адресов (Network Address Translation, NAT), называемое также сокрытием или маскарадингом IP-адресов (*IP masquerading*), дает возможность изменять IP-адреса и номера портов при прохождении пакетов через цепочку. Чаще всего с помощью NAT осуществляют доступ к Интернету нескольких систем в закрытой сети через один IP-адрес. NAT – сложная тема, которую мы не будем подробно обсуждать, но приведем один простой пример в конце данной главы. Подробнее о NAT можно узнать из «NAT HOWTO» или из книги «TCP/IP. Network Administration».[1]

Подсчет пакетов и байтов

В *netfilter* есть счетчики, позволяющие получать статистику применения каждого правила к сетевому трафику, и несколько основанных на ней систем учета. Счетчики можно увидеть при выводе наборов правил командой *iptables* в режиме подробного отчета; мы покажем это в примере 26.3 далее в этой главе.

Использование команды iptables

С помощью команды *iptables* вносятся изменения в цепочки и наборы правил *netfilter*. Она позволяет создавать новые цепочки, удалять цепочки, выводить правила цепочки, очищать цепочки (то есть удалять все имеющиеся в цепочке правила) и устанавливать действие по умолчанию для цепочки. Команда *iptables* также дает возможность вставлять, добавлять, удалять и заменять правила в цепочке.

---

[1] Крэйг Хант «TCP/IP. Сетевое администрирование», 3-е издание. – Пер. с англ. – СПб.: Символ-Плюс, 2005.

У команды *iptables* большое число аргументов и параметров командной строки, но достаточно использовать ее несколько раз, и синтаксис становится вполне очевидным. В этом разделе мы расскажем только о самых частых способах применения *iptables*, поэтому некоторые аргументы и параметры в последующем изложении опущены. В частности, мы не обсуждаем здесь цепочки, определяемые пользователем. В табл. 26.1 перечислены аргументы *iptables*, действующие над цепочками, а в табл. 26.2 перечислены аргументы *iptables*, действующие над отдельными правилами.

Таблица 26.1. Операции iptables над цепочками

Аргумент	Описание
`-L` *цепочка*	Выводит список правил в данной цепочке или во всех цепочках.
`-F` *цепочка*	Удаляет правила из заданной цепочки или из всех цепочек.
`-Z` *цепочка*	Обнуляет счетчик байтов в заданной цепочке или во всех цепочках.
`-P` *цепочка действие*	Определяет действие по умолчанию для заданной цепочки.

Таблица 26.2. Операции iptables над правилами

Аргумент	Описание
`-A` *цепочка спецификация-правила*	Добавляет правило в цепочку.
`-D` *цепочка номер-правила*	Удаляет из цепочки правило с номером *номер-правила*.
`-R` *цепочка номер-правила спецификация-правила*	Замещает спецификацию правила с номером *номер-правила*.
`-I` *цепочка номер-правила спецификация-правила*	Вставляет в цепочку правило с номером *номер-правила* и с заданной спецификацией. Если номер не задан, предполагается 1.

Каждое правило фильтрации содержит параметры, описывающие пакеты, соответствующие правилу. Наиболее употребительные параметры правил приведены в табл. 26.3. Восклицательный знак (`!`) перед параметром инвертирует его. Например, параметр `-dport 80` означает «соответствие порту назначения 80», а параметр `-dport ! 80` означает «соответствует любому порту назначения, кроме 80».

Таблица 26.3. Параметры правил iptables

Параметр	Описание
`-p !` *протокол*	Протокол пакета. Допускаются значения `tcp`, `udp`, `icmp` или `all`.
`-s !` *источник/маска*	Адрес источника пакета, указанный как имя хоста или IP-адрес. Маска задает необязательную маску сети литералом или количеством битов. Например, `/255.255.255.0` задает маску сети литералом, `/24` определяет количество единичных ведущих битов в маске.
`-d !` *источник/маска*	Адрес получателя пакета. Синтаксис такой же, как в адресе источника.

Параметр	Описание
`--sport ! порт`	Порт источника пакета. Задается как номер порта или имя службы из */etc/services*.
`--dport ! порт`	Порт получателя пакета. Синтаксис такой же, как в порте источника.
`-i ! интерфейс`	Сетевой интерфейс, через который получен пакет.
`-o ! интерфейс`	Сетевой интерфейс, через который пакет будет отправлен.

Ряд важных параметров, используемых при создании наборов правил, перечислен в табл. 26.4.

Таблица 26.4. Другие важные параметры iptables

Параметр	Описание
`-v`	Подробный вывод. Удобно использовать для вывода правил с ключом *-L*.
`-n`	Отображать IP-адреса в числовой форме (то есть без поиска имени хоста в DNS).
`-m имя_модуля`	Загрузить расширение *iptables* с именем *имя_модуля*.

Помимо параметров соответствия в каждом правиле *netfilter* должно быть указано действие, совершаемое над пакетом, соответствующим правилу. Обычно в правиле указывается, что пакет должен быть принят или отброшен, как описывается ниже. Если для правила не задано действие, счетчики пакетов и байтов для этого правила наращиваются, а пакет передается следующему правилу в цепочке. Благодаря этому можно создавать правила, используемые только для статистики. При задании действия для правила применяется синтаксис:

```
-j target
```

Здесь `-j` происходит от слова «jump» («переход») и означает, что, если пакет соответствует этому правилу, происходит переход к действию с именем *target*. Значение *target* может быть следующим:

`ACCEPT`

Разрешить передачу или прием пакета.

`DROP`

Отбросить пакет.

`QUEUE`

Передать пакет на обработку программе в пространстве пользователя.

`RETURN`

При использовании в цепочке, определенной пользователем, осуществляется возврат пакета в «вызвавшую» цепочку. При использовании во встроенной цепочке пакет перебрасывается в ее конец, где над ним выполняется действие, установленное для цепочки по умолчанию.

При использовании параметра *-j* можно задать в *target* имя цепочки, определенной пользователем, что дает возможность определять «вложенную цепочку»

правил для обработки этого пакета. Как уже говорилось, *target* со значением RETURN используется для возврата пакета из цепочки, определенной пользователем, в «вызвавшую» цепочку.

Создание наборов правил для IP-фильтрации

Часто при реализации IP-фильтрации труднее всего решить, чего вы действительно от нее хотите. Можно ли разрешить беспрепятственное установление исходящих соединений? Следует ли разрешить перемещение пакетов ICMP? Какие сервисы UDP требуются? Что нужно регистрировать в журналах?

Большая проблема при создании правил фильтрации связана с тем, что большинству пользователей непривычно думать на языке адресов, протоколов и номеров портов. Мы чаще думаем в терминах приложений и конечных пользователей. Чтобы строить правила фильтрации, нужно научиться переводить наши требования на высоком уровне в детали низкого уровня, на котором действует фильтрация.

Некоторое понимание того, как работают сервисы, которыми вы управляете с помощью IP-фильтрации, просто необходимо. В первую очередь нужно знать, какой из протоколов TCP или UDP использует сервис и на каком порте. В файле */etc/services* можно часто найти много нужной информации. Например, при поиске в этом файле smtp вы обнаружите tcp/25, что говорит об использовании протоколом SMTP порта TCP с номером 25. Аналогично при поиске DNS можно найти две записи, в одной из которых указано udp/53, а в другой – tcp/53; это означает, что сервис использует порт 53 как с протоколом TCP, так и с протоколом UDP.

У некоторых протоколов, например FTP, есть две различных записи в */etc/services*. Как говорилось выше, FTP использует один порт для передачи команд (tcp/21) и другой – для передачи данных (tcp/20). К сожалению, клиенты и серверы FTP могут выбирать любые порты для передачи данных, поэтому FTP представляет собой некоторую проблему для правил фильтрации. К счастью, *netfilter* обеспечивает помощь благодаря функции, называемой *отслеживанием состояния соединения* (*connection tracking*), и вспомогательному модулю, ориентированному на сервис FTP. Благодаря этому необходимо создать правило только для передачи команд FTP, а *netfilter* автоматически найдет и разрешит работу соединения, осуществляющего передачу данных. Мы продемонстрируем это позднее в примере 26.2.

Если информации в */etc/services* окажется недостаточно, можно почитать соответствующий документ RFC, в котором описывается протокол, используемый службой. Обычно о сервисе требуется знать лишь то, какие протоколы и порты он использует, что можно легко выяснить в RFC.

Управление IP-фильтрацией и файлы сценариев

Правила фильтрации хранятся и используются ядром аналогично записям маршрутизации: при перезагрузке системы правила IP-фильтрации настраиваются заново. Чтобы гарантировать восстановление настройки брандмауэра при перезапуске системы, необходимо поместить соответствующие команды *iptables* в файл сценария, который автоматически выполняется системой во время загрузки. В связке с программным пакетом *iptables* поставляются две программы с имена-

ми *iptables-save* и *iptables-restore*, которые записывают текущие настройки *netfilter* в файл и восстанавливают их из этого файла, соответственно. Эти средства значительно облегчают задачу управления настройкой брандмауэра.

Каждый дистрибутив обладает своим, слегка отличающимся подходом к управлению настройками брандмауэра:

Red Hat (версии 7.0 и выше)

Сначала установите правила IP-фильтрации с помощью соответствующих команд *iptables*. Затем выполните команду:

```
/sbin/service iptables save
```

В результате правила фильтрации будут записаны в файл */etc/sysconfig/iptables*, автоматически считываемый во время начальной загрузки.

Debian

Для настройки правил *iptables* необходимо либо самостоятельно написать сценарий, размещаемый в каталоге */etc/init.d*, либо воспользоваться одним из пакетов, осуществляющих создание правил брандмауэра.

SUSE Linux

Простую, но не слишком гибкую конфигурацию можно создать, запустив *yast2* и выбрав модуль настройки брандмауэра Security&Users→Firewall.

Другой способ:

- Отредактируйте */etc/sysconfig/SuSEfirewall2*. Этот файл подробно документирован.
- При необходимости определите пользовательские правила фильтрации в */etc/sysconfig/scripts/SuSEfirewall2-custom*. Для этого нужно углубленное понимание того, как правила фильтрации действуют в Linux.
- Запустите фильтрацию командой */sbin/SuSEfirewall2 start*.

Примеры настройки netfilter

В этом разделе мы приведем некоторые простые, но достаточно интересные примеры настройки IP-фильтра. Цель состоит не в том, чтобы дать набор решений, которыми можно безоговорочно воспользоваться. Мы хотим показать, как выглядит работоспособный набор правил фильтрации IP, и дать основу для создания ваших собственных конфигураций.

Пример простого IP-фильтра

Сейчас мы продемонстрируем базовое использование IP-фильтрации, аналогичное тому, как мы использовали TCP-обертки выше в этой главе. Мы хотим отфильтровать пакеты от любых хостов в Интернете, за исключением пакетов, предназначенных демону finger от небольшой группы хостов. Хотя для выполнения той же функции могут быть использованы TCP-обертки, IP-фильтр позволяет проверять многие различные типы пакетов (например, пакеты ICMP ping) и часто необходим, чтобы защитить сервисы, не управляемые TCP-обертками.

В отличие от TCP-оберток, правила *iptables* не могут использовать имена хостов для указания источника или получателя пакетов[1]: при задании правил нужно пользоваться IP-адресами. Однако это даже хорошо, потому что обратный поиск имени хоста не вполне надежен для идентификации пакета (можно фальсифицировать DNS, представив, будто у некоторого IP-адреса другое имя хоста). В примерах 26.1 и 26.2 вместо имен хостов использованы IP-адреса, которые можно получить с помощью такой утилиты, как *host*.

Пример 26.1. Простой пример ipchains

```
# Загрузить модули отслеживания соединений, если они не скомпилированы в ядре.
modprobe ip_conntrack
modprobe ip_conntrack_ftp

# Установить DROP политикой по умолчанию для цепочки INPUT.
iptables -P INPUT DROP

# Принимать пакеты, принадлежащие существующему соединению.
# '-A INPUT' для добавления правила в цепочку INPUT.
# '-m state' использует модуль stateful inspection.
iptables -A INPUT -m state --state ESTABLISHED,RELATED -j ACCEPT

# Принимать все пакеты, поступившие от петлевого интерфейса,
# то есть от локального хоста. '-i lo' определяет петлевой интерфейс.
iptables -A INPUT -i lo -j ACCEPT

# Принимать новые входящие соединения и пакеты, принадлежащие
# существующим соединениям, на порт 22 (ssh).
iptables -A INPUT -m state --state NEW -m tcp -p tcp \
         --dport 22 -j ACCEPT

# Принимать новые входящие соединения FTP от 192.168.1/24.
iptables -A INPUT -m state --state NEW -m tcp -p tcp -s 192.168.1/24 \
         --dport 21 -j ACCEPT

# Принимать новые входящие соединения FTP от spaghetti.vpizza.com,
# у которого IP-адрес 10.21.2.4.
iptables -A INPUT -m state --state NEW -m tcp -p tcp -s 10.21.2.4 \
         --dport 21 -j ACCEPT

# Принимать новые входящие соединения FTP от *.vpizza.com.
# Там две сети: 172.18.1.0 и 172.25.3.0.
iptables -A INPUT -m state --state NEW -m tcp -p tcp -s 172.18.1/24 \
         --dport 21 -j ACCEPT
iptables -A INPUT -m state --state NEW -m tcp -p tcp -s 172.25.3/24 \
         --dport 21 -j ACCEPT
```

Представленный набор правил принимает все пакеты, принадлежащие существующему соединению. Это требуется в случае FTP, когда клиент и сервер могут договориться об альтернативном порте для соединения, в котором передаются данные. Модуль отслеживания состояния соединений (заданный в правилах с по-

[1] Это не совсем соответствует действительности. Нет ничего криминального, если в правилах на месте IP-адресов будут стоять имена хостов, вот только при таком подходе резко снижается пропускная способность правил (из-за необходимости обращаться к службе DNS за разрешением имен), а потому такой подход не рекомендуется к использованию. – *Примеч. перев.*

мощью *-m state*) удостоверяет, что соединение передачи данных может быть принято.

IP-фильтр для защиты целой сети

В предыдущем примере продемонстрирована настройка IP-фильтра для отдельного хоста. В этом разделе мы рассмотрим случай, когда некоторая сеть машин (например, все машины в доме или небольшом офисе) подключена к Интернету через машину-шлюз. Мы можем написать правила *netfilter* для фильтрации трафика между Интернетом и внутренней сетью. В данном случае мы помещаем правила в две цепочки – `INPUT` и `FORWARD`. Напомним, что `INPUT` используется для фильтрации входящих пакетов, адресованных данному хосту, а `FORWARD` применяется к пакетам, пересылаемым шлюзом (то есть адресованным внутренней сети или Интернету). Мы предполагаем, что машина-шлюз использует для связи с Интернетом интерфейс `ppp0`.

Пример 26.2. Использование netfilter для защиты сети IP

```
# Загрузить модули отслеживания соединений, если они не скомпилированы в ядре.
modprobe ip_conntrack
modprobe ip_conntrack_ftp

# Установить DROP политикой по умолчанию для цепочек INPUT и FORWARD.
iptables -P INPUT DROP
iptables -P FORWARD DROP

# Разрешать все пакеты, поступившие от петлевого интерфейса.
iptables -A INPUT -i lo -j ACCEPT

# Создать новую цепочку, определенную пользователем. Она будет содержать
# правила для INPUT и FORWARD, поэтому, объединив их в одной цепочке,
# мы избегаем их дублирования.
iptables -N allowfwdin

# Принимать пакеты, принадлежащие существующему соединению.
# Обратите внимание, что это (и последующие) правила помещены
# в цепочку, определенную пользователем.
iptables -A allowfwdin -m state --state ESTABLISHED,RELATED -j ACCEPT

# Принимать новые запросы соединений от машин во внутренней сети.
# Это позволит машинам внутренней сети устанавливать соединения
# с Интернетом, но не наоборот. Обратите внимание на использование
# `-i ! ppp0` для задания пакетов, поступающих с интерфейсов,
# отличных от ppp0.
iptables -A allowfwdin -m state --state NEW -i ! ppp0 -j ACCEPT

# Принимать новые входящие соединения на порт 22 (ssh).
iptables -A allowfwdin -m state --state NEW -m tcp -p tcp \
         --dport 22 -j ACCEPT

# Принимать новые входящие соединения FTP от 192.168.1/24.
iptables -A allowfwdin -m state --state NEW -m tcp -p tcp -s 192.168.1/24 \
         --dport 21 -j ACCEPT

# Принимать новые входящие соединения FTP от spaghetti.vpizza.com.
iptables -A allowfwdin -m state --state NEW -m tcp -p tcp -s 10.21.2.4 \
         --dport 21 -j ACCEPT
```

```
# Принимать новые входящие соединения FTP от *.vpizza.com.
iptables -A allowfwdin -m state --state NEW -m tcp -p tcp -s 172.18.1/24 \
         --dport 21 -j ACCEPT
iptables -A allowfwdin -m state --state NEW -m tcp -p tcp -s 172.25.3/24 \
         --dport 21 -j ACCEPT

# Ко всем пакетам, прошедшим цепочку, определенную пользователем, теперь
# применяется действие LOG, которое их регистрирует.
# Используется модуль 'limit', чтобы заблокированные пакеты
# не регистрировались слишком быстро.
iptables -A allowfwdin -m limit --limit 2/sec -j LOG

# Установить DROP действием по умолчанию для цепочки, определенной пользователем.
iptables -A allowfwdin -j DROP

# Направлять все пакеты, полученные для INPUT или FORWARD, в нашу цепочку,
# определенную пользователем.
iptables -A INPUT -j allowfwdin
iptables -A FORWARD -j allowfwdin

# Включить IP-маршрутизацию (требуется для всех маршрутизаторов IP,
# независимо от использования IP-фильтра).
echo 1 >/proc/sys/net/ipv4/ip_forward
```

Чтобы проследить все попытки проникнуть через систему защиты, мы добавили правило, регистрирующее пакеты, которые будут отброшены. Однако при поступлении большого количества плохих пакетов это правило могло бы вызвать переполнение диска записями в журнале или недопустимое замедление работы шлюза (поскольку регистрация пакета отнимает гораздо больше времени, чем пересылка или фильтрация). Поэтому мы используем модуль *limit*, контролирующий частоту применения действия правила. В предшествующем примере мы разрешили регистрировать в среднем два плохих пакета в секунду. Все остальные пакеты проскочат правило и будут просто отброшены.

Чтобы посмотреть на правила, которые мы создали (см. пример 26.3), воспользуемся *iptables* с параметром *-L*. Режим подробного вывода (*-v*) показывает больше информации, чем базовая команда вывода.

Пример 26.3. Вывод правил iptables для примера 26.2

```
# iptables -L -v
Chain INPUT (policy DROP 0 packets, 0 bytes)
 pkts bytes target     prot opt in     out     source          destination
   16  1328 ACCEPT     all  --  lo     any     anywhere        anywhere
    0     0 allowfwdin all  --  any    any     anywhere        anywhere

Chain FORWARD (policy DROP 0 packets, 0 bytes)
 pkts bytes target     prot opt in     out     source          destination
    0     0 allowfwdin all  --  any    any     anywhere        anywhere

Chain OUTPUT (policy ACCEPT 9756 packets, 819K bytes)
 pkts bytes target      prot opt in     out      source          destination

Chain allowfwdin (2 references)
 pkts bytes target      prot opt in     out      source          destination
    0     0 ACCEPT      all  --  any    any      anywhere        anywhere \
```

```
state RELATED,ESTABLISHED
0     0 ACCEPT      all  --  !ppp0 any      anywhere        anywhere \
state NEW
0     0 ACCEPT      tcp  --  any   any      anywhere        anywhere \
state NEW tcp dpt:ssh
0     0 ACCEPT      tcp  --  any   any      192.168.0.0/24  anywhere \
state NEW tcp dpt:ftp
0     0 ACCEPT      tcp  --  any   any      10.21.2.4       anywhere \
state NEW tcp dpt:ftp
0     0 ACCEPT      tcp  --  any   any      172.18.0.0/24   anywhere \
state NEW tcp dpt:ftp
0     0 ACCEPT      tcp  --  any   any      172.25.0.0/24   anywhere \
state NEW tcp dpt:ftp
0     0 LOG         all  --  any   any      anywhere        anywhere \
limit: avg 2/sec burst 5 LOG level warning
0     0 DROP        all  --  any   any      anywhere        anywhere
```

Пример сокрытия IP-адресов

С помощью правил *netfilter* можно осуществлять сокрытие IP-адресов (IP masquerading) – особый вид NAT, при котором пакеты из внутренней сети изменяются таким образом, что выглядят как исходящие из одного IP-адреса. Это часто используется, когда есть ряд машин, объединенных в локальную сеть, и одна из машин подключена к Интернету с единственным IP-адресом. Такая ситуация встречается в домашних сетях, когда провайдер выделяет один IP-адрес. С помощью сокрытия IP-адресов целая сеть машин может совместно использовать этот адрес. В результате того, что шлюз осуществляет сокрытие IP-адресов, все пакеты из внутренней сети будут выглядеть так, будто исходят непосредственно от машины-шлюза, а пакеты из Интернета будут пересылаться обратно нужному хосту во внутренней сети. Добиться этого можно с помощью определенной хитрой перезаписи пакетов с помощью *netfilter*.

Настроить *netfilter* для поддержки сокрытия IP-адресов значительно проще, чем объяснить, как это работает! Более полная информация о том, как осуществляются сокрытие IP-адресов и NAT, имеется в «NAT HOWTO». Самый простой вариант настроек приводится в примере 26.4.

Пример 26.4. Базовая конфигурация для сокрытия IP-адресов

```
# Загрузить модуль поддержки NAT, если он не скомпилирован с ядром.
modprobe iptables_nat

# Скрыть IP-адреса для всех маршрутизируемых соединений,
# поддерживаемых устройством ppp0.
iptables -t nat -A POSTROUTING -p ppp0 -j MASQUERADE

# Включить IP-маршрутизацию.
echo 1 >/proc/sys/net/ipv4/ip_forward
```

Здесь предполагается, что есть Linux-система, действующая как шлюз для внутренней сети. У шлюза имеется PPP-соединение с Интернетом через интерфейс `ppp0` и соединение с локальной сетью через интерфейс `eth0`. Предлагаемая конфигурация разрешает исходящие соединения из внутренней сети с Интернетом, но блокирует входящие соединения из Интернета с машинами внутренней сети, за

исключением шлюза. Оказывается, для такого режима не нужны специальные команды, поскольку он действует по умолчанию, когда NAT используется подобным образом.

В этом наборе правил нужно отметить некоторые важные детали. Функциональность NAT обеспечивается отдельным модулем, который нужно загрузить, если он не встроен в ядро. Модуль NAT использует новую цепочку с именем `POSTROUTING`, которая обрабатывает пакеты после того, как ядро выполнит над ними операции маршрутизации (то есть определит, предназначен пакет для Интернета или для машин локальной сети). Параметр `MASQUERADE` определяет все операции по трансляции адресов и отслеживанию пакетов.

Обратите внимание, что в этой конфигурации не происходит фильтрации исходящих соединений. Все хосты частной сети смогут устанавливать исходящие соединения с любыми хостами и любыми портами. В документе «packet filtering HOWTO» есть полезная информация о том, как совместить в одном наборе правил IP-фильтрацию и преобразование сетевых адресов.

SELinux

SELinux – достаточно новая разработка в области обеспечения безопасности Linux. Она была разработана в Управлении национальной безопасности (NSA) Соединенных Штатов Америки с целью обеспечения безопасности компьютерных систем и средств связи в Соединенных Штатах. Любопытно, что правительственное управление, суть которого как раз и состоит в том, чтобы вторгаться в чужие компьютеры и перехватывать чужой трафик, занялось разработкой системы Linux, которая должна быть способна противостоять подобным вторжениям. Во всех подробностях SELinux описывается в книге «SELinux» (O'Reilly).

SELinux – это измененное ядро Linux, обеспечивающее механизм принудительного управления доступом, а также множество утилит, с помощью которых производится управление новыми функциональными возможностями ядра. В системе SELinux пользовательские программы (и демоны) получают доступ ровно к такому количеству ресурсов, в каком они нуждаются. Бреши в системе защиты, такие как переполнение буфера в веб-сервере, больше не могут поставить под угрозу всю систему. В SELinux нет пользователя root, который обладает неограниченными привилегиями.

К сожалению, в рамках данной книги невозможно описать порядок установки и повседневной работы в SELinux, но если вас эта тема заинтересовала, обращайтесь на сайт *http://www.nsa.gov/selinux*. Дополнительные сведения о том, как установить ядро SELinux в некоторых дистрибутивах, можно найти на сайте *http://selinux.sf.net*.

27

Резервное копирование и восстановление

После чтения предыдущих трех глав у вас есть все навыки, необходимые для того, чтобы приступить к работе в системе. Но, в конечном счете, информация данной главы вам тоже понадобится. Есть ряд важных видов работ, которые должны войти у вас в привычку, в том числе создание резервных копий. Также может оказаться полезным умение организовать доступ к файлам и программам Windows. И, наконец, мы поможем вам справиться с происшествиями, которые, будем надеяться, никогда не произойдут, но иногда все же случаются – паникой ядра системы и нарушением целостности данных.

Создание резервных копий

Создание резервных копий системы является важным способом защиты от повреждения или потери данных при возникновении проблем с аппаратным обеспечением или совершении таких ошибок, как непреднамеренное уничтожение важных файлов. При работе с Linux вы, вероятно, сделаете многочисленные настройки системы, которые нельзя будет восстановить, просто переустановив систему с исходного носителя. Однако если исходные диски CD-ROM или DVD-ROM находятся под рукой, необходимости делать резервную копию системы целиком может и не возникнуть. Исходный инсталляционный носитель уже является превосходной резервной копией.

Работая в Linux, как и во всякой UNIX-подобной системе, будучи зарегистрированным как *root*, можно совершить ошибку, в результате которой станут невозможными загрузка системы или регистрация в ней. Часто новички решают эту проблему, целиком переустанавливая систему из резервной копии или, что еще хуже, «с нуля». Однако в действительности в этом редко возникает необходимость. В разделе «Действия в аварийных ситуациях» мы поговорим о том, что делать в таких случаях.

Если вы действительно потеряли данные, иногда их можно восстановить с помощью средств сопровождения системы, описанных в разделе «Проверка и исправление файловых систем» главы 10. Однако, в отличие от некоторых других операционных систем, обычно оказывается невозможным восстановить файл, удаленный командой *rm* или перезаписанный поверх опрометчиво выполненной командой *cp* или *mv* (например, копирование одного файла поверх другого уничтожает

последний). В этих крайних случаях резервные копии являются ключом к решению возникших проблем.

Резервные копии обычно записываются на магнитные ленты, гибкие диски, CD-R(W) или DVD-R(W). Никакой носитель не является надежным на все 100%, хотя в долгосрочном плане ленты, CD-R(W) и DVD-R(W) более надежны, чем дискеты. В наше время, когда стоимость жестких дисков постоянно снижается, а емкость увеличивается, их также можно использовать в качестве носителей резервных копий.

Есть много программ, облегчающих изготовление резервных копий. В простейшем случае можно использовать сочетание *gzip* (или *bzip2*) и *tar* для сохранения файлов с жесткого диска на дискете или ленте. Это самый хороший способ, если резервные копии делаются достаточно редко, скажем, не чаще раза в месяц.

Если на вашей машине много пользователей или вы часто изменяете параметры настройки, то более разумно использовать инкрементную схему резервного копирования. Такая схема предполагает, что полную резервную копию системы вы создаете примерно раз в месяц. Затем каждую неделю вы сохраняете только те файлы, которые изменились за последнюю неделю. Аналогично каждый вечер вы можете сохранять лишь те файлы, которые изменились за последние сутки. Есть несколько программ, которые облегчают резервное копирование такого типа.

В основе инкрементной схемы лежит мысль, что более эффективно осуществлять резервное копирование небольшими шагами. При этом используется меньше дискет, лент или компакт-дисков, а еженедельные и ежедневные резервные копии меньше по размеру и их легче делать. Благодаря такому подходу вы всегда будете иметь резервную копию не более чем однодневной давности. Если, скажем, вы случайно удалите всю систему целиком, то можно восстановить ее из резервной копии следующим образом:

1. Восстановите последнюю ежемесячную копию. Например, если вы затерли систему 17 июля, восстановите ее из полной резервной копии от 1 июля. Теперь ваша система находится в том состоянии, в каком она была в момент создания резервной копии 1 июля.
2. Восстановите все еженедельные резервные копии, сделанные в течение месяца. В нашем случае можно восстановить две еженедельные копии от 7 и 14 июля. Восстановление каждой еженедельной резервной копии обновляет все файлы, изменившиеся в течение соответствующей недели.
3. Восстановите все ежедневные копии за последнюю неделю. В нашем случае должны быть восстановлены ежедневные резервные копии за 15 и 16 июля. Теперь система находится в том состоянии, в каком она была при создании резервной копии 16 июля; потеряны файлы не более чем за один день.

В зависимости от размера системы для сохранения полной ежемесячной резервной копии может потребоваться 4 Гбайт пространства или более. Часто для этого нужно не больше пространства, чем умещается на один DVD-R(W), но достаточно много ZIP-дисков. Однако для еженедельных и ежедневных резервных копий требуется обычно значительно меньше места. Характер использования вашей системы может позволить вам создавать еженедельную копию вечером в воскресенье и не утруждать себя ежедневными копиями по выходным дням.

Важная характеристика, которой обычно должны обладать резервные копии, – это возможность выбирать для восстановления отдельные файлы. Благодаря такой возможности при случайном удалении файла или группы файлов они могут быть восстановлены без необходимости проводить полное восстановление системы. В зависимости от того, как вы создаете резервные копии, такая задача может выполняться очень просто или с мучительными сложностями.

Кроме того, желательно, чтобы носители с резервными копиями физически находились отдельно от компьютера. Если предполагается сохранять резервные копии на жестком диске, рассмотрите вариант внешнего носителя USB, FireWire или SCSI. Предпочтительнее использовать внешние приводы USB и FireWire, потому что они легко подключаются или отключаются от системы по мере необходимости. Если предполагается хранить резервные копии на втором жестком диске, находящемся внутри системного блока, тогда по крайней мере нужно держать этот диск размонтированным, когда он не используется, чтобы в случае непредвиденной ошибки файловая система с резервными копиями не была удалена. Не менее существенно определиться с оценкой важности ваших данных и с тем, насколько соответствует стоимость носителя для хранения резервных копий ценности этих данных.

В этом разделе мы намерены обсудить использование *tar*, *gzip* и некоторых других нужных программ для создания резервных копий на CD и лентах. Мы даже расскажем о том, как использовать CD-R и ленты. Эти средства позволяют делать резервные копии в основном «вручную». Этот процесс можно автоматизировать, написав сценарии оболочки, и даже осуществлять создание резервных копий по расписанию в ночное время с помощью демона *cron*. Все, что от вас требуется, – это вовремя вставлять ленты в механизм. Есть другие программные пакеты для создания резервных копий, восстановления из резервных копий отдельных файлов и т. д. с прекрасными управляемыми меню интерфейсами. Многие из них являются на самом деле удобными оболочками для *tar* и *gzip*. Вы сами должны решить, какой тип системы создания резервных копий вам больше подходит.

Простые резервные копии

Простейший способ создания резервной копии – это использование *tar* для архивирования всех файлов системы или только файлов, расположенных в избранных каталогах. Однако, прежде всего, следует решить, какие файлы сохранять. Следует ли сохранять резервную копию каждого файла системы? В этом редко возникает необходимость, особенно если у вас есть исходные инсталляционные дискеты или CD-ROM. Если вы сделали в системе существенные изменения, но все остальное сохранилось таким, каким оно было на инсталляционном носителе, то можно обойтись архивированием только тех файлов, в которые вы внесли изменения. Хотя с течением времени уследить за такими изменениями становится все труднее.

Обычно производятся изменения в файлах с настройками, находящихся в каталоге */etc*. Есть и другие файлы с настройками, и не помешает заархивировать такие каталоги, как */usr/lib* и */etc/X11* (содержащий файлы с настройками XFree86, как мы видели в разделе «Установка X.org» главы 16).

Следует также сделать резервную копию исходных текстов кода ядра (если вы его обновили или скомпилировали собственное ядро), который находится в каталоге */usr/src/linux*.

Развлекаясь с Linux, неплохо вести учет изменений, которые вы произвели в системе, чтобы обдуманно делать резервные копии. В параноидальном случае можно сделать и резервную копию всей системы; это обойдется вам не дороже, чем стоимость используемых носителей.

Конечно, надо сделать и резервные копии домашних каталогов всех пользователей системы. Обычно они находятся в каталоге */home*. Если в вашей системе есть электронная почта (см. раздел «Postfix MTA – агент передачи почты» главы 23), то может понадобиться сохранить файлы поступившей почты всех пользователей. Часто старые, но важные почтовые сообщения пользователи хранят в буфере входящей почты, и эти файлы нетрудно случайно повредить из-за ошибки почтовой программы или по иной причине. Обычно эти файлы расположены в каталоге */var/spool/mail*. Конечно, это относится только к локальной почтовой системе, а не к тем, кто использует для доступа к почте POP3 или IMAP.

Создание резервной копии на магнитной ленте

Если вы уже решили, какие файлы и каталоги нужно резервировать, можно приступать к делу. Можно непосредственно использовать команду *tar* для создания резервных копий, как было показано в разделе «Использование tar» главы 12. Например, команда

```
tar cvf /dev/qft0 /usr/src /etc /home
```

архивирует все файлы из каталогов */usr/src*, */etc* и */home* на */dev/qft0*. */dev/qft0* является первым устройством «флоппи-ленты», то есть ленточного привода, подключенного к контроллеру дисковода гибких дисков. Этим интерфейсом пользуются многие распространенные ленточные приводы для PC. Если у вас есть накопители на магнитной ленте с интерфейсом SCSI, то они именуются */dev/st0*, */dev/st1* и т. д., в зависимости от номера устройства. Ленточные устройства с другими типами интерфейсов именуются иначе. Эти имена можно выяснить в документации к драйверу устройства, включенному в ядро.

Считать обратно архив с ленты можно командой

```
tar xvf /dev/qft0
```

Это ничем не отличается от работы с *tar*-файлом на диске, демонстрировавшейся в разделе «Утилиты архивирования и сжатия» главы 12.

При использовании ленточного привода лента рассматривается как поток, читать из которого или записывать в который можно лишь в одном направлении. После того как *tar* закончит работу, устройство магнитных лент будет закрыто, а лента перемотана обратно. На ленте не создается файловая система, также не нужно монтировать ее как файловую систему или пытаться обращаться к данным на ленте как к файлам. Воспринимайте ленточное устройство как один архивный «файл».

Магнитные ленты должны быть отформатированы, прежде чем ими можно будет воспользоваться. При форматировании на ленту записываются маркер начала ленты и сведения о сбойных блоках. Для форматирования лент QIC-80 (с ко-

торыми работают драйверы флоппи-лент) можно воспользоваться утилитой *ftformat*, которая либо содержится в дистрибутиве, либо может быть загружена с сайта *ftp://sunsite.unc.edu/pub/Linux/kernel/tapes* в составе пакета *ftape*.

Использование ленты для хранения лишь одного *tar*-архива может оказаться расточительным, если этот архив занимает незначительную часть емкости ленты. Чтобы разместить на ленте несколько архивов, нужно избегать перемотки ленты в начало после каждого ее использования, а также иметь возможность подводить ленту к очередному маркеру файла как при создании архива, так и при его распаковке.

Это можно осуществить при использовании устройств без перемотки ленты в начало; они называются */dev/nqft0*, */dev/nqft1* и т. д. для флоппи-лент и */dev/nst0*, */dev/nst1* и т. д. – для ленточных устройств с интерфейсом SCSI. Если такие устройства используются для чтения или записи, лента не перематывается в начало при закрытии устройства (то есть по завершении работы *tar*). Затем можно снова воспользоваться *tar* для добавления на ленту еще одного архива. Два *tar*-файла на одной ленте никак не связаны между собой. Конечно, если вы затем перепишете первый *tar*-файл, то можете затереть при этом второй или оставить между первым и вторым файлом нежелательный промежуток, который будет восприниматься как мусор. Обычно нужно стремиться к тому, чтобы не переписывать отдельные файлы, если на ленте их несколько.

При использовании ленточного устройства, не перематывающего ленту в начало, можно записать на нее столько файлов, сколько позволит емкость ленты. Возвратить ленту в начало после работы можно командой *mt*. *mt* является командой общего назначения, выполняющей с ленточным устройством ряд операций. Например, команда

```
mt /dev/nqft0 rewind
```

возвращает в начальное положение ленту на первом устройстве флоппи-лент.

Аналогично команда

```
mt /dev/nqft0 reten
```

снимает механические напряжения в ленте путем перемотки ее в конец, а затем обратной перемотки в начальное положение.

При чтении файлов с ленты, содержащей несколько файлов, нужно использовать устройство без перемотки ленты назад, а также команды *tar* и *mt* для позиционирования ленты на нужный файл.

Например, перейти к следующему файлу на ленте можно командой

```
mt /dev/nqft0 fsf 1
```

При этом на ленте пропускается один файл. Чтобы пропустить два файла, нужно выполнить команду

```
mt /dev/nqft0 fsf 2
```

или

```
mt device fsf 1
```

чтобы перейти к следующему файлу.

Команду *mt* нужно использовать с соответствующим устройством, не перематывающим ленту назад. Обратите внимание, что эта команда не перемещает ленту к «файлу номер 2», а пропускает очередные два файла относительно текущего положения ленты. Если вы точно не знаете, в каком положении лента находится в данный момент, отмотайте ее назад командой *mt*. Можно пропускать файлы и в обратном направлении. Полный список параметров команды вы найдете на странице справочного руководства по *mt*(1).

Создание резервных копий на CD-R

Создание резервных копий файлов на компакт-дисках выполняется намного проще, чем на магнитных лентах. Чистые компакт-диски стоят очень недорого, они широко распространены, могут читаться практически в любой системе и намного удобнее в транспортировке и хранении, чем магнитные ленты. В этом разделе описываются самые основы процедуры создания резервных копий на компакт-дисках, а также некоторые интересные приемы, используемые при этом. Практически все, что рассказывается в этом разделе, в равной степени применимо и к дискам DVD.

Безусловно, наиболее типичный способ записи данных на компакт-диск заключается в создании образа диска в виде файла на жестком диске, который затем записывается на болванку компакт-диска. Сделать это достаточно просто, но у такого подхода есть один недостаток: для создания образа компакт-диска в файловой системе нужно иметь по крайней мере 650–700 Мбайт свободного пространства. В современных системах это не является большой проблемой.

На компакт-дисках используется файловая система стандарта ISO 9660, которая может монтироваться и читаться практически в любой современной операционной системе. Создать такую файловую систему можно с помощью инструмента *mkisofs*, после чего ее можно будет использовать тем или иным способом, включая запись на компакт-диск. Фактическая запись образа на болванку в Linux выполняется с помощью другого инструмента – *cdrecord*. Обе эти программы обычно входят в состав большинства систем Linux.

Ниже приводится пример, где сначала создается образ ISO 9660, а затем он записывается на компакт-диск. Предположим, что имеется каталог */data*, который необходимо записать на компакт-диск:

```
# mkisofs -T -r -o /tmp/mycd.iso /data
# cdrecord -v -eject -fs=4M speed=8 dev=0,0,0 /tmp/mycd.iso
```

Некоторые из параметров команды *cdrecord* зависят от используемой системы. Чтобы определить устройство записи компакт-дисков, установленное на машине, можно запустить команду *cdrecord -scanbus*. На компьютере, использовавшемся для проверки сведений к этому разделу, устройство записи компакт-дисков было обнаружено как устройство 0,0,0. Даже несмотря на то, что у автора имеется привод для записи компакт-дисков на скорости 52X, была выбрана скорость записи 8X, чтобы избежать возможного переполнения внутренних буферов устройства и не испортить болванку. Поэкспериментировав со своим устройством, вы сможете самостоятельно определить, до какого скоростного порога запись выполняется надежно.

Существует и другой, хотя и менее надежный способ записи данных на компакт-диск, без создания промежуточного образа. Для этого достаточно объединить команды *mkisofs* и *cdrecord* в конвейер:

```
mkisofs -T -r /data | cdrecord -v -eject -fs=4M speed=8 dev=0,0,0 -
```

Но это не единственный способ оптимизации. Если по каким-то причинам возникает необходимость рассматривать компакт-диск как магнитную ленту, можно вообще пропустить этап создания файловой системы ISO 9660 и сразу записывать архив *tar* прямо на компакт-диск. Такой диск нельзя будет смонтировать или прочитать в системе Windows, но этот способ может оказаться для вас предпочтительнее:

```
tar -czf - /data | cdrecord -v -eject -fs=4M speed=8 dev=0,0,0 -
```

Важно отметить, что хотя качество современных устройств записи заметно выросло за последние годы, тем не менее все еще существует вероятность создать никуда не годный диск, если поток данных, поступающий в устройство записи, прервется, пусть даже на мгновение. Эта проблема особенно проявляется, когда данные подаются на вход *cdrecord* по конвейеру, как в последних двух примерах. Поэтому призываем вас: всегда проверяйте качество сделанной резервной копии!

Кроме *cdrecord*, *tar* и *mkisofs* существует огромное число программ, доступных через Web, которые предоставляют упрощенный интерфейс к утилитам создания резервных копий. Некоторые из них способны создавать резервные копии на нескольких компакт-дисках или управлять ротацией дисков CD-RW. Если описанные здесь приемы создания резервных копий на CD не удовлетворяют ваших потребностей, возможно, что программа, которая вам подойдет, уже была написана кем-то.

Создание резервных копий на жестких дисках

Жесткие диски становятся все больше и все дешевле, единственная проблема состоит в несоответствии между их объемами и объемами носителей резервных копий. Ныне, когда появились жесткие диски объемом 500 Гбайт и стали доступны обычным людям, может случиться так, что для создания резервной копии такого большого диска потребуется... такой же большой жесткий диск!

Практически все приемы резервного копирования на любые носители в принципе могут использоваться и для создания резервных копий на жестких дисках. Однако есть несколько моментов, которые следует рассмотреть.

Если у вас имеется диск, смонтированный в каталог */data*, и вам необходимо создать его резервную копию на другом жестком диске (равном или превышающем исходный диск по размеру), смонтированном в каталоге */backup*, можно выполнить простую перезапись данных, объединив в конвейер две команды *tar*:

```
cd / ; tar -cvf - /data | (cd /backup ; tar -xf -)
```

Если на диске с резервной копией останется свободное место, его можно использовать для хранения инкрементных резервных копий, воспользовавшись методикой, описанной выше в этой главе. Самое замечательное в использовании жестких дисков для хранения резервных копий состоит в том, что они позволяют создавать иерархию каталогов, наилучшим образом отвечающую вашим потребностям. Так, можно было бы смонтировать жесткий диск с резервной копией

в каталог */backup* и пользоваться подкаталогами */backup/full* и */backup/incremental* или любой другой структурой каталогов по своему желанию.

Комбинируя утилиты *find*, *cron*, *tar* и *gzip*, можно создать чрезвычайно короткий, но очень мощный сценарий, который монтировал бы диск с резервной копией, упаковывал бы файлы, изменившиеся с момента создания предыдущей копии, удалял бы резервные копии, сделанные до момента создания последней полной резервной копии, и затем размонтировал бы диск с резервной копией.

Сжимать или не сжимать?

Есть веские аргументы как в пользу, так и против компрессии *tar*-архивов при создании резервных копий. Общая проблема состоит в том, что при всем удобстве использования как *tar*, так и утилиты компрессии *gzip* и *bzip2* не очень устойчивы в отношении возможных ошибок. Сжатие с помощью *gzip* или *bzip2* может значительно сократить объем носителей, необходимых для сохранения архива, однако сжатие целых *tar*-файлов при выводе на CD-R или ленты порождает опасность потерять всю копию при повреждении лишь одного блока архива в результате, скажем, ошибки на носителе (что для CD-R и магнитных лент не такая уж редкость). Большинство алгоритмов сжатия, в том числе *gzip*, зависят от согласованности данных на протяжении больших участков. При повреждении каких-либо данных в сжатом архиве *gunzip* может оказаться вообще не в состоянии декомпрессировать файл, что сделает его полностью нечитаемым для *tar*.

Это значительно хуже, чем если бы *tar*-файл находился на ленте в несжатом виде. Хотя *tar* не обеспечивает большой защиты от повреждения данных в архиве, но если повреждения данных невелики, то обычно можно без особых трудностей извлечь большую часть упакованных файлов, по крайней мере, до того места, где произошло повреждение. Это не идеально, но лучше, чем полная потеря резервной копии.

Для изготовления резервных копий более эффективно использовать вместо *tar* специальные средства архивации. Есть выбор из нескольких вариантов. *cpio* является утилитой архивации, упаковывающей файлы сходным с *tar* образом. Однако благодаря использованию более простого способа хранения данных *cpio* корректно восстанавливает данные из поврежденных архивов. (Ошибки в сжатых файлах она все же обрабатывает не очень хорошо.)

Наилучшим решением может быть использование такой программы, как *afio*. *afio* поддерживает многотомные архивы и во многих отношениях сходна с *cpio*. Однако *afio* осуществляет сжатие и более надежна, поскольку каждый файл сжимается отдельно. Благодаря этому при повреждении архива ущерб может быть ограничен отдельными файлами вместо утраты всей резервной копии.

Эти программы должны быть в вашем дистрибутиве Linux или их можно получить с любого находящегося в Интернете архива Linux. Ряд других утилит резервного копирования с разной степенью популярности и удобства разработан или перенесен на Linux. Если вас серьезно беспокоит проблема создания резервных копий, следует посмотреть эти утилиты.[1] В число программ такого рода входят свободно распространяемые *taper*, *tob* и *Amanda*; есть также коммерческие

[1] Кстати, этот раздел был написан после того, как автор сделал резервную копию своей системы Linux впервые за почти четыре года работы!

программы, такие как *ARKEIA* (бесплатна при использовании не более чем на двух компьютерах), *BRU* и *Arcserve*. Массу бесплатных утилит копирования можно найти на сайте *http://www.tucows.com/downloads/Linux/ISIT/FileManagement/BackupRestore*.

Инкрементное резервное копирование

Инкрементное резервное копирование, описанное ранее в этой главе, является хорошим способом поддержания обновленных резервных копий. Например, можно каждую ночь делать резервные копии файлов, которые изменились за последние сутки, еженедельные копии файлов, изменившихся в течение недели, и ежемесячные копии всей системы.

Инкрементные копии можно создавать с помощью упомянутых выше средств: *tar*, *gzip*, *cpio* и других. При создании инкрементной копии нужно сначала создать список файлов, изменившихся в течение некоторого промежутка времени. Это легко сделать с помощью команды *find*.[1] Если пользоваться специальной программой для создания резервных копий, то, скорее всего, вам понадобится не *find*, а установка какого-то параметра, указывающего на желание сделать инкрементную резервную копию.

Например, чтобы создать список файлов, измененных в течение последних 24 часов, можно использовать команду

```
find / -mtime -1 \! -type d -print > /tmp/filelist.daily
```

Первым аргументом *find* является каталог, с которого нужно начать поиск. В данном случае это корневой каталог /. Параметр *-mtime -1* указывает на то, что нужно найти все файлы, измененные за последние сутки.

Параметр *\! -type d* является сложным (и необязательным), но он удаляет из списка, получающегося на выходе, некоторые ненужные сведения. Он извещает *find* о необходимости исключить из результирующего списка каталоги. Восклицательный знак является оператором отрицания (который в данном случае означает «исключить файлы типа d»), но перед ним нужно поставить символ обратного слэша, поскольку иначе командная оболочка будет интерпретировать его как специальный символ.

Параметр *-print* вызывает вывод имен файлов, удовлетворивших критерию поиска, на стандартное устройство вывода. Мы перенаправляем стандартный вывод в файл для последующего использования. Аналогично для поиска всех файлов, измененных за последнюю неделю, можно воспользоваться командой:

```
find / -mtime -7 -print > /tmp/filelist.weekly
```

Обратите внимание, что при таком использовании *find* эта команда осуществляет поиск во всех смонтированных файловых системах. Если, например, у вас смонти-

---

[1] Если вы еще незнакомы с *find*, постарайтесь поскорее это сделать. *find* предоставляет прекрасный способ найти среди разных каталогов файлы с определенными именами, правами доступа или временем модификации. *find* может даже выполнить заданную программу для каждого найденного файла. В общем, *find* – это ваш друг, и всем грамотным системным администраторам хорошо известно, как ею пользоваться.

рован CD-ROM, *find* будет пытаться найти файлы и на нем (хотя вы, возможно, не собираетесь делать резервную копию CD-ROM). Чтобы ограничить круг поиска локальными файловыми системами, можно воспользоваться параметром *-xdev*. Можно также выполнить *find* несколько раз, указывая при этом первый аргумент, отличный от /. Подробности вы найдете на страницах руководства по *find*(1).

Теперь у вас есть список файлов для включения в резервную копию. Раньше при использовании *tar* мы указывали архивируемые файлы в командной строке. Однако список файлов может быть слишком велик, чтобы уместиться в одной командной строке, которая обычно ограничена 2048 символами, да и сам список хранится в файле.

Можно воспользоваться параметром *-T*, чтобы указать *tar* на файл, содержащий список подлежащих архивированию файлов. Чтобы применить этот параметр, нужно использовать для *tar* альтернативный синтаксис, в котором все параметры задаются явно с дефисами. Например, для архивирования файлов, перечисленных в */tmp/filelist.daily*, на устройство */dev/qft0* выполните команду

```
tar -cv -T /tmp/filelist.daily -f /dev/qft0
```

Теперь можно написать коротенький сценарий, автоматически создающий список файлов и их резервную копию с помощью *tar*. С помощью демона *cron* можно выполнять этот сценарий каждую ночь в определенное время; вам нужно будет позаботиться только о том, чтобы в приводе находилась магнитная лента. Аналогичные сценарии можно написать для создания еженедельных и ежемесячных копий. О *cron* рассказывается в главе 10.

Действия в аварийных ситуациях

Работая в системе с привилегиями *root*, легко совершить простую ошибку, в результате которой возникнут серьезные проблемы, например невозможность регистрации или утрата важных файлов. Это особенно часто случается с неопытными администраторами, начинающими исследовать свою систему. Почти все начинающие системные администраторы приобретают свои знания ценой суровых уроков, когда им приходится восстанавливать систему в чрезвычайной ситуации. В этом разделе мы дадим несколько советов, как действовать, когда неизбежное произойдет.

Необходимо знать меры профилактики, ослабляющие воздействие таких чрезвычайных ситуаций. Например, нужно создавать резервные копии всех важных системных файлов, если не всей системы. Если ваш дистрибутив Linux находится на CD-ROM, то последний служит прекрасной резервной копией большинства файлов (если только в затруднительном положении у вас сохранился доступ к CD-ROM, о чем будет сказано ниже). Резервные копии имеют жизненно важное значение при восстановлении во многих случаях. Не дайте пропасть многим неделям тяжкого труда по настройке системы.

Кроме того, ведите учет системной конфигурации, записывая содержимое таблицы разделов, размеры и типы разделов, файловые системы. Если вы каким-то образом разрушили таблицу разделов, иногда можно решить проблему, просто запустив *fdisk*. Однако это возможно только в том случае, если вы знаете, как выглядела ваша таблица разделов. (История из жизни: с одним из авторов такое

однажды случилось в результате загрузки с чистой дискеты, когда содержание таблицы разделов записано не было. Нет нужды объяснять, что над восстановлением таблицы разделов пришлось поломать голову.)

На самом деле будет совсем не лишним создать резервную копию таблицы разделов для каждого жесткого диска в вашей системе. Программа *sfdisk* – очень удобный инструмент для просмотра, сохранения и изменения информации о разделах. С ее помощью можно получить эту информацию и сохранить ее в файле, запустив следующую команду:

```
sfdisk -d > /partitions.lst
```

Она выведет таблицы разделов всех дисков в системе и сохранит их в файле */partitions.lst* (или с любым другим именем по вашему выбору). Результаты работы *sfdisk* читаемы не только для человека, но и для самой программы *sfdisk*. Если потребуется восстановить таблицу разделов, тогда можно будет удалить из файла */partitions.lst* все таблицы, которые вы *не* собираетесь восстанавливать, и перестроить таблицу разделов (например, для устройства *hda*):

```
sfdisk /dev/hda < partitions.lst
```

Конечно, для того чтобы эти приемы могли работать, необходимо иметь возможность в аварийной ситуации загрузить систему и получить доступ к файлам или восстановиться с резервных копий. Лучше всего сделать это, воспользовавшись аварийным диском или корневым диском. Обычно это загружаемый CD-ROM, содержащий все необходимое для восстановления файловых систем Linux и многих других аварийных работ. Кроме того, существуют такие компакт-диски, как Knoppix (*http://http://www.knopper.net/knoppix/index-en.html*), которые загружают систему с графическим рабочим столом, веб-броузером и всем остальным, что пригодится для комфортной работы. В тяжелой ситуации вам пригодится диск любого из этих типов.

Для систем, которые загружаются исключительно с дискеты, вам потребуется небольшая корневая файловая система с набором инструментов, минимально необходимым для запуска системы Linux с дискеты. Такая дискета используется путем загрузки ядра с другой дискеты (см. раздел «Использование загрузочной дискеты» главы 17) и передачи ядру указания использовать аварийную дискету в качестве корневой файловой системы.

В большинстве дистрибутивов Linux такая комбинация загрузочной и корневой дискет используется в качестве исходной для установки. Установочные дискеты обычно содержат небольшую систему Linux, которая может использоваться для установки программного обеспечения и выполнения основных операций обслуживания системы. Иногда ядро и корневая файловая система находятся на одной дискете, но в этом случае значительно сокращается количество файлов, которые можно записать на аварийную дискету. Польза от таких дискет при обслуживании системы зависит от того, есть ли на них средства (такие как *fsck*, *fdisk*, небольшой редактор типа *vi* и т. п.), необходимые для восстановления системы. В некоторых дистрибутивах процесс установки столь изощренный, что на инсталляционных дискетах остается мало места для чего-либо еще.

Во всяком случае вы можете создать такую корневую дискету сами. Способность сделать это «с нуля» зависит от глубины понимания того, что требуется для загрузки и работы системы Linux, и что можно сократить или отбросить. Напри-

мер, можно оставить в стороне такие запускаемые при старте системы программы, как *init*, *getty* и *login*, если вы знаете, как заставить ядро запустить оболочку на консоли, не прибегая к действительной процедуре начальной загрузки. (Один из способов – на файловой системе дискеты сделать */etc/init* символической ссылкой на */sbin/bash*.)

Не имея возможности раскрыть здесь все детали, отметим, что при создании аварийной дискеты прежде всего необходимо создать на дискете файловую систему с помощью *mkfs* (см. раздел «Создание файловых систем» главы 10). После этого дискета монтируется, и на нее помещаются все необходимые файлы, в том числе нужные элементы из */dev* (большинство из которых можно скопировать из каталога */dev* корневой файловой системы жесткого диска). Вам также понадобится загрузочная дискета, которая содержит просто ядро. С помощью *rdev* нужно установить в этом ядре корневое устройство в */dev/fd0*. Об этом рассказано в разделе «Использование загрузочной дискеты» главы 17. Вы должны также решить, нужно ли загружать файловую систему корневой дискеты на электронный диск, что тоже устанавливается с помощью *rdev*. Если у вас больше 4 Мбайт памяти, то это хорошее решение, поскольку в результате привод гибких дисков можно высвободить для монтирования другой дискеты, содержащей дополнительные программы. Если у вас два привода гибких дисков, это можно сделать без использования электронного диска.

Если после всего сказанного создание аварийной дискеты кажется вам слишком сложным делом, можете попробовать использовать какие-нибудь сценарии, существующие для этой цели (например, *tomsrtbt* на *http://www.toms.net/rb/*). Каким бы способом вы ни создали аварийную дискету, опробуйте ее в действии до того, как несчастье произойдет!

Начать лучше всего с ваших инсталляционных дискет. Если на них нет всех необходимых вам средств, создайте файловую систему на отдельной дискете и поместите на нее недостающие программы. Если вы загружаете корневую систему с дискеты на электронный диск или располагаете вторым дисководом, вы сможете смонтировать эту дискету и получить доступ к своим инструментальным средствам.

Какие средства вам понадобятся? В следующих разделах мы расскажем о наиболее частых аварийных ситуациях и о том, как в этих ситуациях восстанавливать систему; вы узнаете, какие программы требуются в различных ситуациях. Лучше всего, если в помещаемых на дискету программах использована статическая компоновка: это избавит от проблем, вызываемых недоступностью библиотек совместного доступа в аварийной ситуации.

Восстановление файловых систем

Как говорилось в разделе «Проверка и исправление файловых систем» главы 10, в некоторых случаях для восстановления поврежденных файловых систем можно использовать *fsck*. По большей части это касается относительно несерьезных проблем, которые исправляются при обычной перезагрузке системы и запуске *fsck* с жесткого диска. Однако обычно лучше проверять и исправлять не смонтированную файловую систему. В этом случае легче запустить *fsck* с аварийной дискеты.

Разницы между запуском *fsck* с дискеты и с жесткого диска нет: синтаксис совершенно одинаков и описан ранее в этой главе. Учтите, однако, что *fsck* обычно

является интерфейсом для таких программ, как *fsck.ext3*. В некоторых системах нужно использовать *e2fsck* (для вторых расширенных файловых систем).

Повреждение файловой системы может быть таким, что ее не удастся смонтировать. Обычно это связано с повреждением *суперблока* файловой системы, в котором хранится информация о файловой системе в целом. При повреждении суперблока система вообще не сможет установить доступ к файловой системе, и любая попытка монтировать ее закончится отказом (возможно, с сообщением об ошибке «невозможно прочесть суперблок»).

Ввиду важности суперблока файловая система периодически сохраняет его резервные копии. Вторая файловая система разбивается на группы блоков, в каждой из которых по умолчанию 8192 блока. Резервные копии суперблока находятся по адресам со смещением (в блоках) 8193, 16 385 (то есть 8192×2 + 1), 24 577 и т. д. При использовании файловой системы *ext2* или *ext3* можно убедиться, что блоки файловой системы объединены в группы по 8192, выполнив команду

```
dumpe2fs device | more
```

(Разумеется, команда работает, если главный суперблок не поврежден.) Эта команда выводит много сведений о файловой системе, в числе которых вы увидите что-нибудь вроде:

```
Blocks per group:       8192
```

Если будет выведено другое значение, нужно использовать его для вычисления смещений к копиям суперблока, аналогичным показанным выше.

Если вы не можете смонтировать файловую систему из-за проблем с суперблоком, то, вероятно, *fsck* (или *e3fsck*) тоже откажется работать. Для восстановления файловой системы можно предложить *e3fsck* использовать одну из копий суперблока, для чего выполните команду

```
e3fsck -f -b offset device
```

где *offset* является смещением в блоках копии суперблока; обычно равно 8193. Ключ *-f* используется для принудительной проверки файловой системы. При использовании резервных копий суперблока файловая система может показаться «целой», и в этом случае проверка не требуется. Ключ *-f* переопределяет это поведение. Например, для исправления файловой системы на устройстве */dev/hda2* с поврежденным суперблоком можно выполнить команду

```
e3fsck -f -b 8193 /dev/hda2
```

Копии суперблока позволяют спасти положение. Приведенные команды можно выполнить с аварийной дискеты, что может помочь вам снова смонтировать ваши файловые системы.

В последнее время в большинстве дистрибутивов Linux появились так называемые журналируемые файловые системы. Примерами таких систем служат *ext3*, *reiserfs* (файловая система Райзера) и файловая система *jfs*. Эти системы менее подвержены повреждениям, поскольку ведут журнал всех выполненных изменений. Тем не менее все *может* произойти, и потому следует знать, как восстанавливать файловую систему без потери всех файлов, расположенных в ней.

Доступ к поврежденным файлам

При загрузке с аварийной дискеты или CD-ROM вам может потребоваться доступ к файлам на жестком диске. Для этого просто используйте команду *mount*, как описано в разделе «Монтирование файловых систем» главы 10, и монтируйте ваши файловые системы, например, в каталоге */mnt*. (Этот каталог должен существовать в корневой файловой системе аварийного диска.) Например, команда

```
mount -t ext3 /dev/hda2 /mnt
```

позволит получить доступ из каталога к файлам файловой системы *ext3* на устройстве */dev/hda2*. После этого вы можете получить непосредственный доступ к файлам на жестких дисках и даже выполнять с них программы. Например, если вы хотите запустить с жесткого диска редактор *vi*, который обычно находится в каталоге */usr/ bin/vi*, следует выполнить команду

```
/mnt/usr/bin/vi filename
```

Можно даже поместить подкаталоги */mnt* в пути поиска, чтобы облегчить вызов программ.

Не забудьте размонтировать файловые системы на жестких дисках перед перезагрузкой системы. Если ваши аварийные диски не имеют функции корректного останова системы, для верности размонтируйте файловые системы явным образом командой *umount*.

При этом могут возникнуть две проблемы: потеря пароля *root* или искажение содержимого файла */etc/passwd*. В том и другом случае невозможно зарегистрироваться в системе или обрести привилегии *root* командой *su*. Для решения проблемы загрузитесь с аварийного диска, смонтируйте корневую файловую систему в */mnt* и отредактируйте */mnt/etc/passwd*. (Неплохо хранить где-нибудь резервную копию этого файла на случай его непреднамеренного удаления.) Например, чтобы вообще удалить пароль суперпользователя, измените запись для *root* на следующую:

```
root::0:0:The root of all evil:/:/bin/bash
```

Теперь *root* не имеет пароля. Можно перезагрузить систему с жесткого диска и установить пароль командой *passwd*.

Если вас заботят проблемы безопасности системы, вы могли содрогнуться, прочтя предыдущее. Вы поняли правильно: тот, у кого есть физический доступ к вашей машине, может изменить пароль суперпользователя, загрузившись с дискеты. К счастью, есть способы защитить систему от возможной атаки. Более эффективны, конечно, физические средства: если компьютер заперт на замок, никто не доберется до него и не вставит загрузочную дискету. Существуют замки для привода гибких дисков, но учтите – чтобы от них была польза, такая защита должна быть и для привода CD-ROM. Если вы не хотите пользоваться физической защитой, можно воспользоваться паролем BIOS, если ваш компьютер это позволяет: отключите в настройках BIOS попытку загрузки с CD-ROM или дискеты (даже если во время начальной загрузки компакт-диск или дискета вставлены в привод) и защитите настройки BIOS паролем. Это не очень надежно, поскольку пароль BIOS можно сбросить аппаратными средствами, но все же защищает от возможного случайного вторжения. Конечно, существует возможность украсть компьютер целиком.

Другой часто возникающей проблемой является нарушение ссылок на системные библиотеки совместного доступа. Доступ к библиотекам в каталоге */lib* обычно осуществляется через символические ссылки, такие как */lib/libc.so.5*, которые указывают на фактические библиотеки */lib/libc.so.version*. Если ссылка удалена или неверна, многие команды системы не будут исполняться. Проблему можно решить, смонтировав файловые системы на жестких дисках и заново создав ссылки на библиотеки такими командами, как:

```
cd /mnt/lib; ln -sf libc.so.5.4.47 libc.so.5
```

В результате ссылка *libc.so.5* указывает на *libc.so.5.4.47*. Помните, что в символических ссылках используются имена путей, заданные в командной строке *ln*. Поэтому команда

```
ln -sf /mnt/lib/libc.so.5.4.47 /mnt/lib/libc.so.5
```

сделает не то, что нужно: *libc.so.5* будет указывать на */mnt/lib/ libc.so.5.4.47*.

При загрузке с жесткого диска каталог */mnt/lib* будет недоступен, и библиотеку нельзя будет найти. Первая команда работает правильно, поскольку символическая ссылка указывает на файл в том же каталоге.

Восстановление файлов из резервных копий

Если вы удалили важные системные файлы, может оказаться необходимым загрузиться с аварийного диска и восстановить файлы из резервной копии. Поэтому необходимо иметь на аварийных дисках средства для восстановления с резервных копий. В число этих средств входят такие программы, как *tar* и *gzip*, а также драйверы, необходимые для доступа к устройствам, содержащим резервные копии. Например, если резервные копии сделаны на устройстве флоппи-лент, обеспечьте наличие на аварийной дискете модуля *ftape* и команды *insmod*. Подробнее об этом читайте в разделе «Загружаемые драйверы устройств» главы 18.

Для восстановления резервных копий на файловые системы жестких дисков необходимо смонтировать эти файловые системы, как описано ранее, и распаковать содержимое архивов поверх этих файловых систем (например, с помощью соответствующих команд *tar* и *gzip*; см. раздел «Создание резервных копий» этой главы). Не забывайте, что при восстановлении с резервных копий переписываются системные файлы; старайтесь сделать все корректно и не усугубить ситуацию. Большинство программ архивации позволяют извлекать из архива отдельные файлы.

Аналогично, если вы хотите использовать для восстановления файлов свой исходный CD-ROM, нужно иметь в ядре на аварийном диске драйверы, необходимые для доступа к приводу CD-ROM. Тогда можно смонтировать CD-ROM (не забудьте о флагах монтирования *-r -t iso9660*) и скопировать с него файлы.

Файловые системы на аварийных дисках должны также содержать некоторые важные системные файлы. Если какие-либо из них вы удалили из системы, то легко скопировать утраченный файл с аварийной дискеты на файловую систему жесткого диска.

За дополнительной информацией, включая некоторые сценарии, примеры, советы по резервному копированию, обращайтесь к документу «Linux Complete Backup and Recovery HOWTO», написанному Чарльзом Керли (Charles Curley), по адресу *http://www.tldp.org/HOWTO/Linux-CompleteBackup-and-Recovery-HOWTO*.

28

Работа в гетерогенных сетях и запуск программ Windows

Linux – чрезвычайно эффективная операционная система, которая во многих случаях может полностью заменить MS-DOS/Windows. Однако среди нас всегда найдутся такие, кто пожелает использовать наряду с Linux другие операционные системы. На предприятиях, где Linux рассматривается как настольная операционная система, альтернативная Microsoft Windows, часто полагают, что без последней обойтись не удастся, так как имеется необходимость использовать ряд приложений и инструментов, разработанных для работы в среде Win32. Руководители подразделений, отвечающие за развитие информационных технологий, нередко отказываются от использования Linux только потому, что они и не подозревают о возможности запуска приложений Win32 под управлением операционной системы Linux.

Linux удовлетворяет такие сокровенные желания с помощью внутренних расширений, позволяющих получать доступ к посторонним файловым системам и работать с их файлами. Linux имеет возможность монтировать на жестком диске разделы DOS/Windows или обращаться к файлам и принтерам, выделяемым для совместного использования в сети серверами Windows, как это было описано в разделе «Совместный доступ к файлам вместе с Windows (Samba)» главы 15. Linux также может выполнять приложения DOS и Windows с помощью утилит совместимости, позволяющих вызывать MS-DOS или Windows. Кроме того, имеется возможность получать доступ к удаленным системам и запускать программы на этих системах, взаимодействуя с ними с помощью локальной клавиатуры, мыши и экрана монитора.

В этой главе мы используем название «Windows» в некотором общем смысле в отношении любых операционных систем производства Microsoft или совместимых с ними. Хотя операционные системы Windows NT, Windows 2000 и Windows XP существенно отличаются от более старых, основанных на DOS, версий (вплоть до Windows ME), тем не менее большинство инструментальных средств, описываемых в этой главе, будут работать со всеми этими версиями.

Одна из причин, по которым чаще всего возникает потребность в Windows, заключается в том, что нередко она лучше поддерживает новые аппаратные уст-

ройства. Если вы установили Windows из-за того, что вам нужно работать с неким устройством, которое поддерживается Windows, но для которого нет драйвера в Linux, не отчаивайтесь. Большинство основных аппаратных устройств, поддерживаемых Windows, в конечном счете получают поддержку в Linux, хотя, возможно, приходится некоторое время ждать. Например, драйверы Linux для устройств USB одно время были редкими и сырыми, но сейчас многие распространенные устройства USB прекрасно работают в Linux. Самые свежие данные о том, какие устройства USB работают под Linux, можно получить на сайте *http://www.linux-usb.org*.

Windows может потребоваться вам и для работы со «стандартными» приложениями, такими как Photoshop или Microsoft Office. В обоих случаях можно воспользоваться бесплатными свободно распространяемыми приложениями (а именно: The Gimp, KOffice и OpenOffice), которые могут составить конкуренцию и даже превзойти свои фирменные эквиваленты с закрытым программным кодом. Однако иногда все же приходится запускать Windows, чтобы получить доступ к программным продуктам, для которых в Linux нет равноценных или отсутствует полная совместимость.

В сущности, есть четыре способа взаимодействия Linux и Windows:

- Совместное использование CD и дискет («sneakernet»).
- Совместное использование компьютера при установке в разных разделах диска.
- Совместное использование данных в сети.
- Одновременное выполнение на одном компьютере с помощью эмулятора или виртуальной машины.

Когда Windows и Linux работают на разных машинах, не связанных сетью, можно записать дискету или CD (CD-R или CD-RW) на одной машине и прочесть на другой. Как Windows, так и Linux могут читать и записывать CD в стандартном формате ISO 9660. Программа *cdrecord*, работающая под Linux и другими разновидностями UNIX, может создавать CD с использованием расширения стандарта ISO 9660 под названием Joliet, разработанного Microsoft, благодаря чему у Windows не возникает проблем с форматом диска.

Более выгодной экономически является установка Windows и Linux на одной и той же машине в разных разделах жесткого диска. Во время начальной загрузки пользователю дается возможность выбрать операционную систему, которая должна быть запущена. В разделе «Загрузка системы» главы 17 описывается, как настроить загрузку альтернативных операционных систем. Для доступа к файлам на разделах Windows во время работы Linux можно смонтировать раздел Windows непосредственно в файловую систему Linux. В результате появляется возможность обращаться к файлам Windows как к обычным файлам UNIX.

Для компьютеров, объединенных в сеть, самым замечательным средством заставить Linux и Windows работать вместе является пакет Samba – программное обеспечение open source, позволяющее получать доступ к файлам и принтерам UNIX из Windows. Серверы Linux, на которых работает Samba, могут в некоторых случаях обслуживать компьютеры Windows даже быстрее, чем сами серверы Windows! Кроме того, пакет Samba оказался очень стабильным и надежным.

В пакет Samba входят также программы, работающие с файловой системой *smbfs*, поддерживаемой Linux, которая позволяет монтировать в файловой системе Linux каталоги, выделенные Windows для совместного использования. Мы достаточно подробно обсуждали *smbfs* и Samba, чтобы вы могли монтировать совместно используемые каталоги и получить действующий функциональный сервер.

Эмуляторы и виртуальные машины – это разновидность программного обеспечения, которое позволяет запускать приложения Windows и даже саму операционную систему Windows внутри операционной системы Linux. Wine – проект с открытыми исходными текстами, имеющий задачей непосредственную поддержку приложений Windows без установки последней. Другой подход применяется в коммерческом приложении VMware, которое позволяет одновременную работу нескольких операционных систем – Windows, Linux, FreeBSD и некоторых других. При работе Windows под VMware данные используются совместно с хостом Linux с помощью средств Samba.

Наконец, приложения удаленного рабочего стола позволяют пользователям одной системы регистрироваться в другой системе и удаленно запускать приложения и даже администрировать удаленные системы.

Следует проявлять здоровый скептицизм в отношении некоторых заявлений о совместимости. Например, вы можете обнаружить, что необходимо вдвое больше дискового пространства для поддержки двух операционных систем и связанных с ними файлов и прикладных программ плюс средства преобразования файлов и графических форматов и т. д. Может оказаться, что аппаратные устройства, настроенные для работы с одной ОС, не настроены для работы с другой или даже после установки и правильной настройки всего необходимого программного обеспечения остаются небольшие неразрешимые проблемы совместимости.

Совместное использование дисковых разделов

Как описывалось в разделе «Монтирование файловых систем» главы 10, разделы на локальных жестких дисках можно смонтировать непосредственно в файловую систему Linux. Чтобы иметь возможность читать и записывать данные на определенную файловую систему, ее поддержка должна быть включена в ядре Linux.

Файловые системы и монтирование

В Linux есть драйверы для чтения и записи файлов в обычной файловой системе FAT и более новой системе VFAT, которая появилась в Windows 95 и поддерживает длинные имена файлов. Linux может читать (и с некоторыми ограничениями записывать) файловую систему NTFS Windows NT/2000/XP.

В разделе «Сборка нового ядра» главы 18 вы научились собирать собственное ядро. Для доступа к разделам DOS (используется MS-DOS и Windows 3.x) и VFAT (используется Windows 95/98/ME) необходимо включить параметр `DOS FAT fs support` в разделе `Filesystems` во время настройки ядра. После установки этого параметра можно включить параметры `MSDOS fs support` и `VFAT (Windows-95) fs support`. Первый параметр позволяет монтировать разделы FAT, а второй – разделы FAT32.

Если требуется доступ к файлам в разделе Windows NT с файловой системой NTFS, то нужен другой драйвер. Во время настройки ядра включите параметр

`NTFS filesystem support`. Это позволит монтировать разделы NTFS, указав в качестве типа файловой системы *ntfs*. Учтите, что в настоящее время драйвер NTFS поддерживает доступ только для чтения. Существует версия этого драйвера, которая поддерживает и запись, но на момент написания книги она все еще находилась на стадии разработки и не гарантировала надежной работы при записи в раздел NTFS. Внимательно прочтите документацию, прежде чем устанавливать ее и работать с ней!

Во время работы Linux можно монтировать разделы Windows так же, как любые другие типы разделов. Если, например, третий раздел на первом диске IDE содержит установленную копию Windows 98, вы можете сделать его файлы доступными с помощью следующей команды, которую надо выполнить, зарегистрировавшись в качестве *root*:

```
# mount -t vfat /dev/hda3 /mnt/windows98
```

Здесь */dev/hda3* указывает на диск, соответствующий диску Windows 98, а параметр */mnt/windows98* может быть изменен на любой каталог, который вы создали для доступа к этим файлам. Но как узнать, что вам нужен именно */dev/hda3*? Если вы уже знакомы с соглашениями об именах для файловых систем Linux, то знаете, что *hda3* – это главный диск на первом IDE-порте. Можно облегчить себе жизнь, записав имена разделов во время их создания с помощью *fdisk*, но если вы этого не сделали, можно снова запустить *fdisk* и посмотреть на таблицу разделов.

Драйверы файловых систем поддерживают ряд параметров, которые можно задавать с помощью ключа *-o* команды *mount*. Параметры описаны на страницах справочного руководства *mount*(8), отдельные разделы посвящены параметрам, специфичным для файловых систем *fat* и *ntfs*. Раздел с описанием параметров файловой системы `fat`, два из которых представляют особый интерес, относится к обеим файловым системам – *msdos* и *vfat*.

Параметр *check* указывает ядру, что делать с именами файлов, недопустимыми в MS-DOS. Это относится только к созданию и переименованию файлов. Вы можете указать три значения для *check*: `relaxed` позволит сделать с именем файла практически все что угодно. Если имя не отвечает соглашению 8.3 для файлов MS-DOS, оно будет соответствующим образом укорочено. `normal`, являющееся значением по умолчанию, также сократит длинные имена файлов, но, кроме того, удалит специальные символы, такие как `*` и `?`, недопустимые в именах файлов MS-DOS. Наконец, `strict` запрещает и длинные имена файлов, и спецсимволы. Для того чтобы потребовать от Linux строгости по отношению к именам файлов в смонтированном ранее разделе, команду *mount* можно использовать следующим образом:

```
# mount -o check=strict -t msdos /dev/sda5 /mnt/dos
```

Этот параметр применим только к файловым системам *msdos*; ограничения на длину файла не относятся к файловой системе *vfat*.

Второй параметр, *conv*, может оказаться полезен, но не так широко, как покажется на первый взгляд. Системы Windows и UNIX пользуются разными соглашениями относительно того, как указывать конец строки в текстовых файлах. Windows использует и возврат каретки, и символ перевода строки, в то время как UNIX использует только символ перевода строки. Это не значит, что файл из

одной системы нельзя прочесть на другой, но может сильно мешать. Чтобы ядро автоматически выполняло преобразование форматов текстовых файлов Windows и UNIX, необходимо выполнить команду *mount* с параметром *conv*. Здесь возможны три значения: `binary`, являющееся значением по умолчанию, не выполняет никакого преобразования; `text` преобразует все файлы; и `auto` пытается по расширению имени файла определить, является ли файл текстовым или двоичным. Если расширение входит в список «известных двоичных расширений», файл не преобразуется, иначе преобразование выполняется.

Обычно не рекомендуется использовать `text`, так как это приведет к повреждению любых двоичных файлов, включая графические файлы и файлы, написанные в текстовых процессорах, электронных таблицах и других программах. Может оказаться опасным и `auto`, так как механизм распознавания, основанный на расширениях, слишком прост. Поэтому мы не рекомендуем применять параметр *conv*, если только раздел не содержит исключительно текстовые файлы. Устанавливайте режим *binary* (значение по умолчанию) и преобразовывайте файлы по мере необходимости. Ниже в этой главе есть раздел «Утилиты преобразования файлов» с указаниями, как это делать.

Файловые системы MS-DOS и NTFS, как и все остальные, можно автоматически монтировать во время начальной загрузки, для чего нужно поместить в файл */etc/fstab* соответствующую запись. Например, следующая строка в */etc/fstab* монтирует раздел Windows 98 в каталоге */win*:

```
/dev/hda1 /win vfat defaults,umask=002,uid=500,gid=500 0 0
```

При доступе из Linux к одной из файловых систем *msdos*, *vfat* или *ntfs* системе приходится каким-то способом присваивать файлам права доступа и владения, предусматриваемые UNIX. По умолчанию права доступа и владения определяются с помощью UID, GID и маски, устанавливаемой по умолчанию вызывающим процессом. Такой принцип приемлем при выполнении команды *mount* из оболочки, но при выполнении из загрузочных сценариев владельцем файлов будет назначаться суперпользователь, что может быть нежелательно. В приведенном выше примере параметр *umask* устанавливает маску, которую система будет использовать при создании файлов и каталогов. Параметр *uid* задает владельца (в виде числового UID, а не текстового имени), а параметр *gid* задает группу (в виде числового GID). Все файлы на монтируемом разделе будут иметь в Linux владельца и группу. Поскольку системы с загрузкой альтернативных операционных систем обычно используются как рабочие станции, принадлежащие одному пользователю, параметрам *uid* и *gid* можно дать значения UID и GID этого пользователя.

Утилиты преобразования файлов

Одна из самых значительных проблем, возникающих при совместном использовании файлов в Linux и Windows, связана с различием принятых окончаний строк в текстовых файлах. К счастью, есть несколько способов решения этой проблемы:

- Если вы обращаетесь к файлам на смонтированном разделе на той же машине, можно разрешить ядру автоматически преобразовывать файлы, как это описано в разделе «Файловые системы и монтирование» в этой главе. Но пользуйтесь этим аккуратно!

- Стандартные редакторы, такие как Emacs и vi, могут автоматически выполнять преобразование во время загрузки или сохранения файла.
- Имеется большое количество утилит, выполняющих преобразование файлов из одного формата окончания строк в другой. Некоторые из них могут выполнять и другие типы преобразований.
- Напишите собственную утилиту преобразования на том языке программирования, который вам больше нравится.

Если вас интересует только преобразование символов перевода строки, то написать программу для преобразования файлов очень просто. Чтобы преобразовать из формата DOS в формат UNIX, замените все вхождения <CR><LF> (`\r\f` или `\r\n`) в файле на символ перевода строки (`\n`). Для преобразования в обратном направлении замените все символы перевода строки на <CR><LF>. Приведем две программы на Perl, которые решают эту задачу. Первая из них, которую мы назовем *d2u*, преобразует файлы из формата DOS в формат UNIX:

```
#!/usr/bin/perl
while (<STDIN>) { s/\r$//; print }
```

Следующая программа (которую мы назовем *u2d*) преобразует файл из формата UNIX в формат DOS:

```
#!/usr/bin/perl
while (<STDIN>) { s/$/\r/; print }
```

Обе команды читают входной файл со стандартного устройства ввода и пишут выходной файл на стандартное устройство вывода. Легко модифицировать примеры так, чтобы имена входного и выходного файлов задавались в командной строке. Если вам лень писать такие утилиты самостоятельно, можете поискать в своем дистрибутиве Linux программы *dos2unix* и *unix2dos*, которые действуют аналогично нашим простым *d2u* и *u2d*, а также принимают имена файлов в командной строке. Другая аналогичная пара утилит – *fromdos* и *todos*. Если вы не найдете ни одной из них, попробуйте команду *flip*, которая может преобразовывать файлы в обоих направлениях.

Если эти простенькие утилиты кажутся вам недостаточно мощными, можете попробовать *recode*. Эта программа преобразует практически любые стандарты текстовых файлов.

Простейший способ использования *recode* предполагает указание прежнего и нового набора символов (кодировки текстового файла) и файла для преобразования. *recode* заменит старый файл на преобразованный файл с тем же именем. Например, для преобразования текстового файла из Windows в UNIX можно ввести:

```
recode ibmpc:latin1 textfile
```

textfile будет заменен на преобразованную версию. Нетрудно догадаться, что для преобразования этого файла обратно в кодировку Windows надо ввести:

```
recode latin1:ibmpc textfile
```

Помимо кодировки `ibmpc` (используемой в Windows) и `latin1` (для UNIX) имеются и другие возможные варианты, среди которых есть `latex` для кодировки диакритических знаков в стиле LaTeX и `texte` для кодировки почтовых сообщений на французском языке. Полный список можно получить, введя команду:

```
recode -l
```

Если вам не нравится привычка *recode* заменять ваш старый файл новым, можно воспользоваться тем фактом, что *recode* может также читать из стандартного потока ввода и писать в стандартный поток вывода. Для преобразования файла *dostextfile* в *unixtextfile* без удаления *dostextfile* можно выполнить:

```
recode ibmpc:latin1 < dostextfile > unixtextfile
```

С помощью описанных выше утилит можно достаточно комфортно управляться с текстовыми файлами, но это только начало. Например, растровая графика в Windows обычно сохраняется как файлы *bmp*. К счастью, имеются средства преобразования файлов *bmp* в графические файлы формата *png* или *xpm*, которые чаще используются в UNIX. Среди них есть Gimp, который, возможно, входит в ваш дистрибутив.

Ситуация осложняется, когда дело доходит до других файловых форматов, например, используемых в офисных приложениях. В то время как в Windows разные инкарнации формата *.doc*, используемого текстовым процессором Word, превратились де-факто в общепринятый язык текстовых процессоров Windows, до недавнего времени прочесть такие файлы в Linux было почти невозможно. К счастью, недавно появились программы, способные читать (и иногда даже выводить) файлы *.doc*. Среди них офисный пакет KOffice, свободно распространяемый OpenOffice, и коммерческий пакет StarOffice 6.0, близкий родственник OpenOffice. Однако следует знать, что такое преобразование никогда не будет безупречным, и, скорее всего, после него придется редактировать файл вручную. Даже в Windows преобразования не бывают корректными на 100%; если вы попытаетесь импортировать файл Microsoft Word в WordPerfect (или наоборот), то поймете, о чем мы говорим.

Вообще, чем более популярным является формат файла в Windows, тем больше вероятность, что разработчики Linux предоставят способы для чтения или даже записи файлов этого формата. Другой выход – перейти при создании файлов в Windows на открытые файловые форматы, такие как RTF (Rich Text Format) или XML (Extensible Markup Language). В эпоху Интернета, где потоки информации текут свободно, использование закрытых недокументированных файловых форматов – это анахронизм.

Эмуляция и виртуальные операционные системы

Следующий шаг вслед за использованием файлов Windows в среде Linux – это заставить Linux подражать Windows, чтобы появилась возможность запускать приложения Windows. В этом разделе будут обсуждаться два наиболее популярных программных продукта, способных решить эту задачу: Wine (вместе с CrossOver Office) и VMware.

Wine

Wine поможет выбраться из множества щекотливых ситуаций, например, когда ваши друзья досаждают вам предложениями поиграть в последнюю версию Half-Life 2, или когда после перехода вашей организации на Linux вы вдруг узнаете, что ваш начальник не может обойтись без своей любимой базы данных Access.

Wine – это свободно распространяемый пакет программного обеспечения, который позволяет запускать программы Windows под управлением Linux. Достигается это за счет реализации прикладного программного интерфейса Microsoft Win32 (только для платформы Intel x86).

Акроним Wine расшифровывается как «Wine Is Not an Emulator» («Wine – это не эмулятор»). Вместо того чтобы эмулировать Windows, Wine транслирует системные вызовы приложений Windows в системные вызовы Linux. Wine и ее библиотеки можно представить себе как некую программную прослойку, расположенную между приложениями и Linux (мало чем отличающиеся от других API, о которых уже рассказывалось). Однако никто не будет возмущаться, если вы назовете этот продукт эмулятором, потому что он действует аналогичным образом.

Своими корнями Wine уходит в далекий 1993 год, когда операционная система Linux только начинала свое развитие. Группе разработчиков показалось интересным заставить программы Windows работать в Linux. В то время Microsoft использовала прикладной программный интерфейс Win16 в Windows 3.1. Более новая операционная система Windows NT еще находилась в стадии активной разработки, и с ее выходом предполагалось появление широкого диапазона новых технологий, включая прикладной программный интерфейс Win32. Разработчики Wine недооценили объем работ, связанных с необходимостью запуска приложений Win16, а появившиеся в дальнейшем приложения Win32 только добавили новые проблемы. Спустя длительное время наконец-то выяснилось, как следует выстроить архитектуру пакета, чтобы позволить программам Windows работать под Linux. Большая часть ядра Wine была закончена к 2000 году, но еще оставалась нереализованной значительная часть Win32 API – это означало, что потребуется еще несколько лет труда для реализации всех его функциональных возможностей. Последние версии Wine обладают поддержкой дополнительных API, таких как DirectX, Microsoft Installer и COM. В сообществе Wine ходит шутка, что для завершения работы над проектом потребуется от шести до двенадцати месяцев... через десять лет. Однако темпы разработки в последние годы значительно выросли, и вполне возможно, что к тому моменту, когда вы будете читать эти строки, появится стабильная версия Wine.

Однако будем реалистами и отметим, что Wine может дать, а чего не может. Огромное число программ, написанных в последние годы для Windows, будут (что самое приятное) работать под управлением Wine.

Все, что было написано до появления Windows 95, скорее всего, работать не будет. Тому есть масса причин, но самая главная – разделы Wine, реализующие Win16 и DOS API, прекратили свое развитие, и потому ошибки в них остаются неисправленными.

Аналогично приложения, написанные для самых последних версий Windows, могут использовать особенности, еще не реализованные в Wine.

Все, что находится между этими крайностями, а это основная масса приложений, создававшихся для Windows 98, 2000 и XP, будет работать вполне прилично. Наиболее часто в среде Wine используются следующие приложения:

Microsoft Office

Internet Explorer

Adobe Photoshop
Quicken

Вы вполне можете обнаружить, что некоторые категории приложений не будут работать в Wine. В этом случае следует проанализировать, так ли уж нужны эти приложения, а также подумать о возможности приобретения коммерческой версии Wine (будет описана ниже), которая может поддерживать эти приложения.

Борцы за чистоту идеи могут утверждать, что Wine только наносит вред Linux и свободному программному обеспечению. Однако следует признать бесспорным тот факт, что объем существующего программного обеспечения для операционной системы Windows намного превосходит объем программного обеспечения для любой другой операционной системы. Кроме того, ни у кого не вызывает сомнения, что огромное количество программного обеспечения для Windows за долгие годы вышло из употребления, поскольку многие производители ушли из бизнеса. Wine в состоянии расширить перечень поддерживаемого программного обеспечения и помочь вам решить проблемы интеграции, но разработчики Wine будут первыми, кто посоветует в первую очередь попробовать использовать решения, разрабатывавшиеся специально для Linux. Однако, если по каким-то причинам это невозможно, Wine может помочь вам.

Если вы занимаетесь разработкой программного обеспечения, определенный интерес для вас может представлять библиотека Winelib, которая представляет собой версию интерфейсов Win32, экспортируемых для компоновки приложений. Благодаря Winelib вы можете взять исходный текст программы для Windows и пересобрать ее в Linux под управлением Wine. Это дает несколько преимуществ, например возможность запускать программу на аппаратных платформах, отличных от x86. Кроме того, использование Winelib даст возможность вашему приложению обращаться к любой библиотеке Linux. Например, при желании можно было бы интегрировать приложение со звуковой подсистемой Linux, переориентировав требуемые части приложения на работу с ALSA. Приложения для Winelib все еще требуют наличия Wine для управления системными ресурсами, такими как многопоточная среда Windows. Найти Winelib можно на сайте *http://www.winehq.org/site/winelib*.

Wine поможет решить широкий диапазон проблем интеграции, хотя с вашей стороны может потребоваться приложить некоторые усилия, чтобы все заработало достаточно гладко. Не бросайте Wine, если первая же попытка запустить свое любимое приложение не увенчается успехом. Обратитесь к ресурсам, на которые мы укажем здесь, и постарайтесь выяснить, что еще можно сделать для достижения успеха. Проведите некоторое время с инструментами настройки, чтобы изменить значения параметров по умолчанию. Если это не поможет, попробуйте воспользоваться пробной версией CrossOver Office, чтобы выяснить, будет ли приложение работать в этой среде.

Получение и установка Wine

Как и в случае с любым другим свободно распространяемым программным обеспечением, вам придется решить, собирать ли Wine из исходных текстов или воспользоваться скомпилированной версией. Любой из этих подходов имеет свои преимущества, но вам нужно тщательно обдумать свое решение. Мы рекомендуем загрузить скомпилированную версию, если у вас есть такая возможность.

Для сборки из исходных текстов используется стандартный набор инструментов, таким образом, если вы чувствуете себя вполне уверенно, чтобы запустить команды *configure*/*make*, тогда этот вариант установки будет для вас оптимальным. Независимо от того, какой вариант выбран, необходимо будет лишний раз убедиться, что все составные части программного пакета были загружены и настроены в соответствии с параметрами системы.

Пакеты, содержащие как исходные тексты, так и скомпилированную версию, можно найти на сайте проекта WineHQ по адресу *http://www.winehq.org/download*. На этом сайте можно найти дополнительные инструкции по установке в отдельные дистрибутивы Linux, такие как Debian или Ubuntu. При выборе любого пакета вас перенаправят на сайт зеркала для загрузки. Проект Wine развивается очень быстро, поэтому, если в вашей системе имеется предустановленная версия Wine, настоятельно рекомендуем обдумать возможность обновления до последней версии с сайта WineHQ.

Проект Wine поддерживает распространение скомпилированных пакетов для большинства распространенных дистрибутивов Linux, включая Red Hat, Mandriva, Fedora, SUSE, Debian и Slackware. Есть вероятность, что вы без труда найдете пакет, который может быть установлен в ваш дистрибутив. Каждая версия пакета собрана с учетом особенностей отдельных дистрибутивов и может предложить гораздо более тесную интеграцию с операционной системой, чем можно получить при сборке пакета из исходных текстов. Кроме того, сборка двоичных дистрибутивов производится с тем набором библиотек, который поставляется в составе дистрибутивов. Если имеются некоторые сомнения по поводу наличия в вашей системе всего необходимого для сборки Wine из исходных текстов, устанавливайте скомпилированную версию. Для этого можно воспользоваться инструментами управления пакетами, входящими в состав дистрибутива, такими как *rpm* или *apt*.

Если будет принято решение устанавливать Wine из исходных текстов, тогда вам потребуется типичный набор инструментальных средств, необходимых для этого. Wine использует набор стандартных библиотек, входящих в состав любого дистрибутива, но вам придется проверить наличие заголовочных файлов, например от X Window System. Сборка пакета заключается в исполнении серии типичных команд внутри каталога с исходными текстами:

```
$ ./configure
$ make depend
$ make
```

Обратите внимание на результаты работы команды *configure*, чтобы удостовериться, что все необходимые компоненты были найдены. Для установки пакета вам потребуются привилегии суперпользователя. Зарегистрировавшись как *root*, нужно выполнить команду *make install* из каталога с исходными текстами. По умолчанию установка будет выполнена в каталог */usr/local* в такие подкаталоги, как */usr/local/bin* и */usr/local/lib/wine*.

При желании можно выполнить установку, используя самую последнюю версию исходных текстов Wine, загрузив их из репозитория CVS. Дерево CVS с исходными текстами Wine является относительно стабильным, и потому проблемы при сборке весьма маловероятны. Однако, как и в случае с любым другим программным продуктом, над которым ведется работа, вам следует еще раз поду-

мать, так ли нужно находиться на самом острие. Чтобы получить доступ к дереву исходных текстов в CVS, вам нужно будет указать адрес репозитория, зарегистрироваться и затем извлечь исходные тексты:

```
$ export CVSROOT=:pserver:cvs@cvs.winehq.org:/home/wine
$ cvs login
$ cvs -z 0 checkout wine
```

При последующих обновлениях достаточно будет просто перейти в только что созданный каталог *wine* и запустить команду *cvs update -PAd*.

Простой пример использования Wine

Прежде чем начать запускать приложения под управлением Wine, рассмотрим простой пример. Позже мы обсудим настройки, которые были выбраны Wine автоматически. Значений по умолчанию вполне достаточно для запуска простых программ, но потом может потребоваться изменить их, чтобы расширить круг приложений, которые можно будет запускать под Wine.

Начнем прямо сейчас: запустите менеджер задач Wine – приложение *taskmgr*. Если пакет Wine установлен и работоспособен, можно запустить команду:

```
$ wine taskmgr
```

Менеджер задач Wine позволяет запускать, останавливать и отлаживать процессы в Wine.

После запуска *taskmgr* можно заметить некоторые изменения, произошедшие в домашнем каталоге. Параметры настройки Wine сохраняются в каталоге *~/.wine* и во вложенных в него подкаталогах. Внутри *~/.wine* можно обнаружить системный реестр, который был создан и сохранен в виде трех файлов с именами *system.reg*, *user.reg* и *userdef.reg*. Кроме того, в этом каталоге были созданы два каталога, имеющих большое значение: *dosdevices* и *drive_c*. Первый из них содержит все необходимое для настройки виртуальных дисков. Второй содержит виртуальный диск Windows со всеми каталогами, которые обычно создаются операционной системой Windows: *c:\windows* и *c:\Program Files*. О каждом из них мы поговорим подробнее в процессе обсуждения.

Никогда не запускайте Wine с привилегиями пользователя *root*. Программы, запускаемые под управлением Wine, обладают доступом к частям системы в соответствии с привилегиями пользователя, запустившего их. Исполнение программ с привилегиями суперпользователя может привести к появлению проблем с безопасностью и даже разрушить систему. В настройках по умолчанию виртуальный диск Windows отображается в корень файловой системы Linux, таким образом, запуская приложения с привилегиями пользователя *root*, вы предоставляете им возможность изменять или удалять любые файлы в системе.

Настройка Wine

Обычно настройка Wine производится с помощью нескольких инструментов с графическим интерфейсом, но вы можете выполнить ее, вооружившись обычным текстовым редактором. Сердцем механизма настройки Wine является ре-

естр. Поскольку приложения Windows требуют наличия реестра для хранения своих параметров, Wine была вынуждена реализовать его для соблюдения совместимости. На протяжении многих лет Wine поддерживала отдельный файл с настройками (обычный файл *config*). Нет смысла поддерживать два отдельных механизма настройки, поэтому все параметры настройки были перенесены прямо в реестр. Теперь есть возможность настраивать и приложения, и Wine одним набором инструментов.

В составе Wine поставляются два различных инструмента настройки: *winecfg* и *regedit*. Первый из них упрощает процесс настройки параметров общего назначения. Второй позволяет управлять глубинными настройками, а также параметрами настройки приложений. Оба инструмента хранятся в каталоге *~/.wine*. В отличие от Windows, файлы реестра Wine хранятся в обычном текстовом формате. Благодаря этому всегда есть возможность изменить какие-либо параметры с помощью обычного текстового редактора.

Введите в командной строке команду *winecfg*, чтобы запустить приложение. Первое, что бросается в глаза, – множество вкладок в верхней части окна для настройки приложений, библиотек, графической подсистемы, внешнего вида, дисков и аудио. Вкладки Applications (Приложения), Libraries (Библиотеки) и Graphics (Графика) связаны между собой и позволяют управлять настройками отдельных программ. Например, на вкладке Applications (Приложения) можно определить версию Windows, которую Wine будет сообщать приложениям. По умолчанию предполагается эмуляция Windows 2000. Вы можете заметить, что программы ведут себя по-разному, если, к примеру, будет выбрана версия Windows 98 или Windows XP. Если точно известно, что программа требует именно Windows 2000, можно щелкнуть на кнопке Add application (Добавить приложение), отыскать исполняемый файл приложения, а затем изменить версию Windows именно для данного приложения или оставить этот параметр в значении по умолчанию. Все параметры, изменяемые на вкладках Libraries (Библиотеки) и Graphics (Графика), относятся к приложению (или к настройкам по умолчанию), выбранному на вкладке Applications (Приложения).

На вкладке Libraries (Библиотеки) можно изменить поведение отдельных библиотек. Wine позволяет использовать «родные» библиотеки Windows, если таковые имеются. Например, если у вас имеется установленная версия Windows на отдельном разделе, вы можете скопировать библиотеки DLL из каталога *C:\WINDOWS\SYSTEM* в соответствующий каталог на виртуальном диске Windows.

Чтобы подключить эти библиотеки, введите имя библиотеки в текстовом поле под меткой New override for library (Новое замещение для библиотеки). Например, если вы скопировали библиотеку *FOO.DLL* из Windows, введите название `foo` в текстовое поле и щелкните на кнопке Add (Добавить). По умолчанию Wine будет пытаться загрузить «родную» библиотеку Windows только после неудачной попытки загрузить встроенную библиотеку. Щелчком на кнопке Edit (Редактировать) вы можете изменить порядок выбора библиотек: Wine (Встроенная (Wine)), Windows (Родная (Windows)) или комбинация из этих двух вариантов. Однако следует помнить, что библиотеки, составляющие ядро Wine, такие как *kernel32*, *ntdll*, *user32* и *gdi32*, никогда не должны подменяться «родными» библиотеками.

Изменения на вкладке Graphics (Графика) обычно приходится вносить для игровых программ. Здесь, управляя глубиной цвета, можно влиять на производи-

тельность программы. Если X-сервер имеет поддержку расширения XRandR (она имеется в большинстве современных систем), Wine будет пытаться изменить разрешение экрана по запросу приложения. Если такое поведение является нежелательным, Wine может «эмулировать рабочий стол» с размерами по вашему выбору. Виртуальный рабочий стол будет отображаться при запуске приложения в пределах окна X.

Настройки на вкладке Appearance (Внешний вид) не оказывают влияния на работу приложений, но позволяют настроить внешний вид Wine. Здесь можно загрузить темы Windows XP из файлов *.msstyles*.

Программы Windows ограничены в использовании ресурсов файловой системы. С помощью настроек на вкладке Drives (Диски) можно определить, какие файловые системы Linux будут доступны приложениям. Это поможет предотвратить гибельные разрушения файловой системы, если какая-нибудь программа Windows выйдет из-под контроля. Приложения, исполняющиеся в среде Wine, могут изменять только те файлы, которые доступны пользователю Linux, запустившему эти приложения. По умолчанию Wine настраивает специальный диск Windows в каталоге *~/.wine/drive_c* и устанавливает приложения Windows в этот каталог. Если посмотреть на настройки виртуального диска `C:`, можно заметить, что он отображен в этот каталог. Если необходимо добавить еще один диск, нужно щелкнуть на кнопке Add (Добавить) и определить каталог в файловой системе Linux, который будет использоваться для отображения этого диска. Например, если в системе имеется привод для сменных носителей, можно добавить точку монтирования этого устройства как виртуальный диск.

Наконец, название вкладки Audio (Аудио) говорит само за себя. Щелкните на кнопке Autodetect (Автоопределение), и будет выполнена попытка отыскать наиболее подходящий аудиодрайвер Wine. Кроме того, можно выбрать драйвер вручную в раскрывающемся списке. Если у вас возникают проблемы с воспроизведением звука, проверьте настройки системного микшера, возможно, при этом придется подрегулировать громкость воспроизведения. Проблемы со звуковой подсистемой обсуждались в главе 9.

Приложение *winecfg* во многом напоминает панель управления Windows. Она лишь служит графическим интерфейсом для доступа к содержимому реестра. И так же как в Windows, те же самые настройки можно выполнить с помощью инструмента *regedit*. Чтобы запустить это приложение, просто введите в командной строке команду *regedit*. Большинство настроек, которые обсуждались выше, можно найти в ветке реестра `HKEY_CURRENT_USER\Software\Wine`. Например, если залезть поглубже в иерархию реестра с помощью *regedit*, можно заметить, что версия Windows хранится прямо в ветке `HKEY_CURRENT_USER\Software\Wine` в виде параметра `Version`. Кроме того, *regedit* может пригодиться для исследования настроек отдельных приложений. Обычно приложения хранят свои настройки в ветвях реестра, имена которых следуют соглашению `HKEY_LOCAL_MACHINE\Software\`*`vendor`*`\`*`application`*.

Единственные настройки, которые не хранятся в реестре, – это диски и порты. Изменения, выполняемые на вкладке Drives (Диски), сохраняются непосредственно в файловой системе в виде набора символических ссылок. Если заглянуть в каталог *~/.wine/dosdevices*, можно увидеть, что каждая ссылка, соответствующая виртуальному диску, указывает на вполне определенный каталог в файло-

вой системе. По умолчанию конфигурация виртуальных дисков выглядит примерно следующим образом:

```
lrwxrwxrwx 1 wineuser wineuser   10 May 6 21:37 c: -> ../drive_c
lrwxrwxrwx 1 wineuser wineuser    1 May 6 21:37 z: -> /
```

Здесь виртуальный диск `c:` ссылается на каталог *~/.wine/drive_c*, а виртуальный диск `z:` – на корень файловой системы. Доступ к портам определяется аналогичным образом, например, в случае необходимости можно добавить символическую ссылку с именем *com1*, указывающую на файл устройства */dev/ttyS0*. Конечно же, Wine обладает возможностью доступа лишь к тем частям файловой системы, которые доступны самому пользователю.

Запуск Wine

Фактический запуск приложения выполняется командой *wine*. Прежде всего, необходимо отыскать файл приложения, который требуется запустить. В первую очередь, это означает, что необходимо отыскать файл установки приложения. Обычно инсталляционный файл имеет имя *SETUP.EXE* или *INSTALL.EXE*. Чтобы запустить его, нужно ввести команду:

```
$ wine SETUP.EXE
```

После этого должна запуститься программа установки и скопировать все необходимые файлы на виртуальный диск Windows. Кроме того, программа выполнит все необходимые настройки в реестре. В Windows это может включать изменение некоторых специальных ключей реестра, таких как `RunOnce`, который содержит список программ, исполняемых при перезагрузке Windows. Приложения, описываемые ключом `RunOnce`, запускаются всего один раз, но обычно они выполняют необходимые операции по подготовке программы к использованию. Чтобы эмулировать перезагрузку Windows и дать возможность этим приложениям отработать, нужно запустить команду *wineboot*. Эта команда ничего не может повредить в системе, поэтому ее всегда необходимо выполнять по окончании процесса установки программы.

После того как программа будет установлена, необходимо отыскать исполняемый файл и запустить его. Существует вероятность, что Wine изменит параметры настройки рабочего стола и создаст элемент запуска установленной программы, таким образом вы можете увидеть ярлык на своем рабочем столе. Кроме того, в главном меню рабочего стола может появиться новый пункт, соответствующий новой программе. Wine будет стараться создавать свое специализированное подменю и размещать все ссылки на установленные программы в нем.

Если этого не происходит или вы предпочитаете запускать приложения из командной строки, перейдите в каталог, куда было установлено приложение. Скорее всего, это будет каталог *~/.wine/drive_c/Program\ Files/application*. Большинство Windows-приложений запускаются из рабочего каталога, куда эти приложения были установлены. Находясь в этом каталоге, просто введите команду:

```
$ wine program.EXE
```

В этот момент вы можете обнаружить, что необходимо выполнить дополнительные настройки параметров с помощью *winecfg*, чтобы определить версию Windows или заместить библиотеку DLL для данной программы. Если установка

Wine производилась из исходных текстов, обратите внимание на сообщения, начинающиеся со слова FIXME, которые сообщают о нереализованных функциях Windows. Нередко появление таких сообщений спровоцировано проблемами в библиотеках. Замещение библиотек «родными» для Windows может решить эти проблемы. Кроме того, может помочь установка программного обеспечения сторонних производителей. Проблемы, связанные с выходом в Интернет, нередко решаются установкой Internet Explorer и библиотек, поставляющихся вместе с ним.

Убедитесь, что устанавливаете лицензионные версии программного обеспечения или библиотек.

Если приложение работает в текстовом режиме, его также можно запустить с помощью Wine. В этом случае можно изменить драйвер графического устройства с *winex11.drv* на *winetty.drv*, тогда приложение будет запускаться прямо в окне терминала, не создавая отдельного окна X Window. Однако некоторые текстовые программы ведут себя не совсем корректно и требуют наличия графических возможностей, поэтому они могут не запускаться без поддержки X Window. Запустить такую программу можно с помощью *wineconsole*:

```
$ wineconsole program.EXE
```

В перечень других программ, которые можно запускать с помощью *wine*, входят Wine-версии приложений *Notepad*, *REGEDIT* и многих других программ, которые обычно имеются в составе Windows. Wine-версии этих программ используют в своей работе Winelib, и все элементы графического интерфейса, которые вы увидите на экране, отображаются средствами Wine.

Дополнительная справочная информация

Если вам придется столкнуться с какими-либо проблемами, посетите сайт *http://www.winehq.org*, где можно найти множество дополнительных сведений. Здесь имеются значительные объемы документации для пользователей. Ответ на ваш вопрос может содержаться в списке часто задаваемых вопросов (FAQ). В противном случае следует обратиться к руководству пользователя Wine (Wine User Guide), где дается более глубокий охват тем, обсуждаемых здесь, а также ряд практических рекомендаций. Наконец, на сайте можно найти сборник документов, разоблачающих расхожие мифы и описывающие принцип действия Wine, а также рассказывающие о некоторых особенностях.

Кроме того, существует несколько ресурсов, которые помогут разрешить проблемы, связанные с Wine. Если заглянуть в базу данных приложений (Application Database) по адресу *http://appdb.winehq.org*, можно обнаружить, что кому-то уже удалось разрешить проблему, с которой вам пришлось столкнуться. При желании можно отправить отчет об ошибке в базу данных Wine Bugzilla по адресу *http://bugs.winehq.org*. Помимо этого в поиске решения проблемы может помочь список рассылки, имеющийся на сайте WineHQ.

Наконец, можно обратиться в интернет-энциклопедию wiki, где сообществом собрана коллекция сведений, по адресу *http://wiki.winehq.org*. При желании здесь вы можете поделиться своими знаниями или отыскать дополнительные документы.

CrossOver Office

Если вам не удалось «победить» Wine, вы можете сделать вывод, что коммерческая версия работает лучше. Компания CodeWeavers ведет разработку пакета CrossOver Office, 30-дневную пробную версию которого можно найти по адресу *http://www.codeweavers.com*. Этот пакет поддерживает небольшой набор Windows-приложений. Попробуйте эту версию, может быть, под ее управлением приложения будут работать.

В CodeWeavers большое внимание уделяется тестированию совместимости Windows-программ со своим продуктом, который обладает по-настоящему хорошей поддержкой этих приложений. В настоящее время список содержит более 30 различных популярных приложений, включая Microsoft Word, Excel, PowerPoint, Visio, Access, Quicken, iTunes, FrameMaker и Lotus Notes. Кроме того, в CrossOver Office можно пользоваться модулями расширения броузера, такими как QuickTime, Shockwave Director и Windows Media Player, а также обеспечить их поддержку в «родном» веб-броузере Mozilla. CrossOver Office предусматривает поддержку как стабильной, так и тестовой версии Wine. Кроме того, компанией CodeWeavers предоставляются консультационные услуги, с помощью которых вы сможете обеспечить работоспособность Windows-приложений в Linux. Если вам нужна надежная, отзывчивая и доброжелательная компания, осуществляющая поддержку Wine, то CodeWeavers отвечает всем этим требованиям.

Продукт CrossOver Office можно заказать и загрузить прямо на веб-сайте CodeWeavers. Имеется возможность загрузить RPM-пакет для Red Hat, SUSE и других дистрибутивов, основанных на RPM, а также специализированный инсталлятор для Debian или инсталлятор на основе Loki. Последний наиболее предпочтителен, потому что он может работать полностью в пространстве пользователя.

После того как инсталлятор Loki будет загружен, необходимо установить бит разрешения на исполнение (*chmod +x*) для сценария установки. После этого достаточно будет просто запустить следующий сценарий:

```
$ ./install-crossover-standard-5.0.sh
```

Сценарий сначала распакует архив CrossOver Office, а затем выполнит установку. По окончании вам будет предоставлена возможность установить Windows-приложения. Для этого будет запущен инструмент настройки *cxoffice/bin/cxsetup*, который представит перечень поддерживаемого программного обеспечения. Мастер установки произведет инсталляцию программного обеспечения и выполнит дополнительные действия, такие как перезапуск Wine, для эмуляции перезагрузки Windows. Кроме того, можно будет установить дополнительное, не поддерживаемое программное обеспечение. В среде CrossOver Office могут работать многие приложения, официально не поддерживаемые, и вы можете обнаружить, что необходимые вам Windows-программы в CrossOver Office работают значительно лучше, чем в обычной среде Wine.

Независимо от того, какое приложение вы пожелаете установить, поддерживаемое или нет, будет предложено указать местоположение инсталляционного файла. Здесь можно выбрать программу установки с CD-ROM или с жесткого диска. Некоторые поддерживаемые приложения устанавливаются автоматически, другие требуют запуска программы установки. После того как будет выполнена установка и произведена эмуляция перезагрузки Wine, в меню окружения рабоче-

го стола появятся новые пункты. В панели быстрого запуска KDE (*Kicker*) или в панели рабочего стола GNOME появится новый элемент с названием Windows Applications, где будут находиться пункты запуска программ, установленных CrossOver Office.

Чтобы настроить CrossOver Office, найдите подменю CrossOver в главном меню рабочего стола и выберите пункт Configuration (Настройка). В результате на экране появится диалог *cxsetup* с вкладками Add/Remove (Добавление/Удаление) и Manage Bottles (Управление комплектами установленного программного обеспечения). На первой из них можно будет управлять приложениями, включая их установку с помощью кнопки Install (Установить), после щелчка на которой запускается мастер установки, описанный выше.

На второй вкладке, Manage Bottles (Управление комплектами установленного программного обеспечения), можно выполнить настройку различных аспектов CrossOver Office. *Bottles* (буквально – бутыль, сосуд) – это автономные, независимые друг от друга каталоги, содержащие комплекты установленных приложений, каждый из которых обладает своими параметрами настройки Wine. После выбора требуемого комплекта из списка и щелчка на кнопке Configure (Настроить) вам будет предоставлена возможность изменить такие параметры настройки, как меню и наборы файловых ассоциаций, настроить модули расширения, добавить шрифты, запустить апплет панели управления и изменить настройки комплекта. На каждой вкладке имеется кнопка Help (Справка) на случай, если вы столкнетесь с затруднениями.

VMware Workstation

VMware (ныне принадлежит EMC) распространяет программные продукты, которые позволяют запускать виртуальные операционные системы на серверах и рабочих станциях.[1] Версия для рабочих станций достигла определенной известности, оказывая помощь тем, кому необходимо запускать различные операционные системы на своем настольном компьютере. Между прочим, этот продукт позволяет устанавливать лицензионные версии Windows в Linux. В процессе разработки VMware Workstation также существенное внимание уделяется тестированию. Многие утверждают, что данный продукт ускоряет процесс развертывания приложений.

Что фактически делает этот продукт?

VMware Workstation 5 позволяет одновременно запускать несколько операционных систем и приложений на одном компьютере. Если у вас достаточно оперативной памяти, приличный объем жесткого диска и современный процессор, VMware будет работать нормально.

На рис. 28.1 приводится внешний вид окна приложения VMware Workstation, работающего под управлением Novell Linux Desktop 9 с установленной Windows XP Home Edition. Из этого рисунка можно получить общее представление о том,

[1] В последнее время набирает популярность еще один инструмент виртуализации – Xen, являющийся свободно распространяемым программным обеспечением и позволяющий запускать на крупных серверах одновременно несколько версий Linux и некоторых других операционных систем.

что можно ожидать от данного продукта. Обратите внимание: на панели инструментов (вверху) имеется кнопка (Full Screen), позволяющая перевести VMware в полноэкранный режим работы. В этом режиме создается полное ощущение, что гостевая операционная система установлена непосредственно на компьютер.

Согласно информации, исходящей от VMware, Workstation 5 позволяет запускать:

32-разрядные системы

SUSE LINUX Pro 9.2, SUSE LINUX Enterprise Server 9.0, Mandrake Linux 10, Red Hat Enterprise Linux 4.0 и Windows Server 2003 SP1 beta (экспериментальная поддержка).

64-разрядные системы

Red Hat Enterprise Linux 4.0, Red Hat Enterprise Linux 3.0, SUSE LINUX Enterprise Server 9, SUSE LINUX Enterprise Server 9 SP1, SUSE LINUX Enterprise Server 8, Windows Server 2003 SP1 (экспериментальная поддержка) и Windows XP (экспериментальная поддержка).

Пользователи Fedora Core 3, Gentoo, Red Hat и Debian отмечают, что им также удалось запустить VMware 5 в их системах без каких-либо затруднений, хотя компания не объявляла о поддержке этих дистрибутивов. Как видно из рис. 28.1, авторы нашли VMware вполне работоспособной в дистрибутиве NLD 9.

Каждая гостевая операционная система запускается внутри изолированной виртуальной машины. VMware отображает аппаратные ресурсы компьютера на ре-

Рис. 28.1. *VMware Workstation 5 в Novell Linux Desktop 9*

сурсы виртуальной машины, благодаря чему каждая виртуальная машина обладает своим собственным виртуальным процессором, памятью, дисками и устройствами ввода-вывода. По крайней мере, так обстоят дела с точки зрения гостевой операционной системы. Для нее виртуальная машина представляется типичным компьютером на базе процессора x86.

После установки VMware Workstation вы можете устанавливать и запускать на одном и том же компьютере немодифицированные версии Windows, Linux, Novell NetWare и Sun Solaris x86, а также приложения, написанные для этих платформ. Программное обеспечение VMware обещает своим пользователям многочисленные преимущества, которые дает использование нескольких компьютеров, при полном отсутствии необходимости обслуживать различные аппаратные платформы. VMware Workstation упрощает пользователям Linux возможность работы с Windows-приложениями.

Может случиться так, что VMware Workstation как программный продукт будет существовать не очень продолжительное время. Безусловно, она предоставляет возможность пользователям Linux запускать Windows-приложения, но при этом они постепенно будут смещаться в сторону использования аналогов этих приложений, которые изначально разрабатывались для использования в GNOME, KDE или других окружениях рабочего стола. Но не стоит забывать, что продукты для рабочих станций – это лишь одно из направлений, в которых работает компания Vmware, и свое будущее эта компания видит в области серверов, где Linux играет главную роль.

Однако VMware занимает важное место в истории развития Linux. Многие до сих пор любят ее продукты и рады возможности пользоваться ими. Хотелось бы надеяться, что компания продолжит развивать свои продукты, упрощающие возможность пользоваться Windows-приложениями для пользователей Linux, пока та часть ее деятельности, что лежит в области серверов, будет расти и крепнуть.

Установка VMware Workstation 5

При установке VMware в SUSE Professional 9.2 нам пришлось столкнуться с некоторыми трудностями, хотя компания причислила этот дистрибутив к поддерживаемым платформам. Процесс установки выглядит достаточно просто. Загрузить дистрибутив VMware можно либо в виде сжатого *tar*-файла, либо в виде пакета RPM. После установки пакета достаточно запустить команду `vmware-config.pl`. Но в SUSE 9.2 эта команда дает следующее сообщение об ошибке:

```
None of the pre-built vmmon modules for VMware Workstation is suitable
for your running kernel. Do you want this program to try to build the
vmmon module for your system (you need to have a C compiler installed
on your system)?
CC [M] /tmp/vmware-config1/vmmon-only/linux/driver.o
/bin/sh: scripts/basic/fixdep: No such file or directory
make[2]: *** [/tmp/vmware-config1/vmmon-only/linux/driver.o] Error 1
make[1]: *** [_module_/tmp/vmware-config1/vmmon-only] Error 2
make[1]: Leaving directory `/usr/src/linux-2.6.11.4-21.7'
make: *** [vmmon.ko] Error 2
make: Leaving directory `/tmp/vmware-config1/vmmon-only'
Unable to build the vmmon module.
```

После этого было принято решение установить Novell Linux Desktop 9 без каких-либо дополнительных обновлений, в результате удалось установить VMware без особых проблем. После этого мы обновили NDL 9 и проверили VMware Workstation 5 – она продолжала работать.

Позднее выяснилось, что на ядра в SUSE 9.x накладывались исправления (патчи, или заплатки), которые отсутствуют в базовом ядре. Многие программные пакеты, в составе которых поставляются прекомпилированные модули, терпят неудачу при установке в систему с этими ядрами. Чтобы как-то решить эту проблему, вы можете пересобрать модули ядра. Для этого нужно выполнить следующий ряд команд:

```
# cd /usr/src/linux
# make cloneconfig
# make prepare
# vmware-config.pl
```

После этого программа *vmvare-config.pl* найдет все требуемые файлы, как если бы она имела дело со свежескомпилированным ядром.

На веб-сайте VMware вас может также обескуражить путаница, касающаяся сведений о том, какие дистрибутивы Linux могут выступать в роли хост-системы, а какие в роли гостевой системы. Представьте себе, что вы пригласили гостя к себе домой. Вы провожаете его в гостиную, усаживаете поудобнее и ведете с ним беседу. В терминах VMware, когда устанавливается виртуальная машина, тем самым создается гостиная, куда можно будет привести гостевую операционную систему.

В этом случае Linux выступает в роли хост-системы, а Windows устанавливается как гостевая. Если вам потребуется установить в качестве гостевой системы другой дистрибутив Linux, наведите справки на веб-сайте VMware, какие дистрибутивы поддерживаются в качестве гостевой операционной системы (в противовес хост-системе). В перечень официально поддерживаемых гостевых операционных систем входят:

Mandrake Linux 8.2, 9.0, 9.2, 10

Novell Linux Desktop 9

Red Hat Linux 7.0, 7.1, 7.2, 7.3, 8.0, 9.0

Red Hat Enterprise Linux AS/ES/WS 4.0 (32-bit)

Red Hat Enterprise Linux AS/ES/WS 2.1, 3.0

Red Hat Enterprise Linux Advanced Server 2.1

SUSE Linux 7.3, 8.0, 8.1, 8.2, 9.0, 9.1, 9.2

SUSE Linux Enterprise Server 7, 7 patch 2, 8, 9, 9 SP1

Turbolinux 7.0, Enterprise Server 8, Workstation 8

Sun Java Desktop System (JDS) 2

Виртуальные аппаратные устройства в VMware 5 функционируют много лучше, чем в предыдущих версиях. Одного процессора Pentium IV или эквивалентного ему процессора AMD и 512 Мбайт оперативной памяти должно быть вполне достаточно для одновременной работы двух виртуальных машин. В предыдущих версиях такое было просто невозможно.

Характеристики VMware Workstation 5

Как только вы привыкнете к необычным ощущениям от возможности пользоваться знакомой операционной системой, находясь в Linux, можно приступать к изучению дополнительных возможностей VMware Workstation 5, которые превращают эту прекрасную платформу в полигон для экспериментов и средство коллективной разработки.

Копии состояния операционной системы. Пятая версия VMware предоставляет возможность создания нескольких копий операционной системы, что позволяет пользователю сохранить образ гостевой системы в некотором состоянии и вернуться к нему позднее после перезапуска VMware. Имеется возможность настроить виртуальную машину так, чтобы создать и сохранить целую вереницу копий состояния системы для последующего исследования. Например, если вам потребуется изучить поведение компьютерного вируса, можно сделать копию до заражения. Если вирус причинит серьезные повреждения, из копии, созданной ранее, можно будет восстановить виртуальную машину до состояния, предшествовавшего заражению. Та же самая возможность может оказаться бесценной в процессе отладки нового программного кода или исправления (патча).

В предыдущих версиях VMware также имелась возможность создания копий, но каждая последующая копия затирала предыдущую. Таким образом, можно считать, что применительно к нуждам тестирования пятая версия получила существенные расширения своих возможностей.

Виртуальные сети. Как уже упоминалось ранее, на одной хост-системе можно запустить несколько виртуальных машин. Эта возможность позволяет пользователям объединить их в виртуальную локальную сеть. Виртуальная сеть настраивается аналогично любой другой локальной сети, с той лишь разницей, что она будет организована внутри единственного компьютера.

Пользователи могут организовать взаимодействие в том, что в терминах VMware называется сегментом локальной сети. Этот сегмент невидим из реальной сети, что предоставляет безопасную среду для ведения разработок.

Клонирование. VMware Workstation 5 обладает интереснейшим механизмом, который компания называет *клонированием* (*cloninng*). В терминах VMware Workstation существуют два типа клонов. Первый называется полным клоном, который можно рассматривать как образ системы, созданный программой *ghost*, готовый для развертывания на другом компьютере. Второй тип называется связанным клоном. Клоны этого типа тесно связаны с оригинальным образом системы.

Полный клон VMware представляет собой полностью независимую копию виртуальной машины. После того как будет создан полный клон, его можно перенести и запустить на другом компьютере независимо от родительской виртуальной машины. Далее его развитие идет уникальным путем, и его можно изменять и распространять в соответствии со своими потребностями.

Связанные клоны используют виртуальное дисковое пространство совместно с оригинальной, или родительской, виртуальной системой, что дает возможность экономить дисковое пространство. Это позволяет использовать одни и те же экземпляры установленного программного обеспечения несколькими виртуальными машинами. Кроме того, на создание связанных клонов уходит гораздо меньше времени, чем на создание полных клонов.

Иногда может потребоваться создать в лабораторных условиях несколько связанных клонов, чтобы предоставить разработчикам, инженерам, тестерам или программистам отдела сопровождения абсолютно идентичную среду окружения. Если связанная виртуальная машина находится в сети, другие пользователи смогут быстро создать связанный клон этой виртуальной машины. Отдел сопровождения может воспроизводить выявленные ошибки на виртуальной машине, а разработчик сможет быстро создать связанный клон этой виртуальной машины для работы над устранением этой ошибки.

Файлы в родительской системе, существовавшие на момент создания связанного клона, остаются доступными для связанного клона. Изменения, производимые в родительской системе, не оказывают влияния на связанный клон, а изменения, производимые на диске связанного клона, никак не влияют на родительскую систему.

Другие программы, позволяющие запускать приложения MS–DOS и Windows в Linux

Был сделан целый ряд попыток решить проблему запуска приложений для MS-DOS и Windows в Linux, как группами разработчиков свободно распространяемого программного обеспечения, так и коммерческими разработчиками. Простейшее решение предоставляет продукт Dosemu (*http://www.dosemu.org*), который достаточно хорошо эмулирует аппаратные средства PC для выполнения MS-DOS (или совместимой системы типа PC-DOS или DR-DOS). DOS все равно требуется устанавливать в эмуляторе, но поскольку DOS фактически выполняется внутри эмулятора, это обеспечивает хорошую совместимость приложений. С некоторыми ограничениями можно выполнять даже Windows 3.1.

Есть другой проект open source под названием Bochs (*http://bochs.sf.net*), который эмулирует аппаратную часть PC достаточно хорошо для работы Windows и других операционных систем. Однако поскольку каждая инструкция 386-го процессора эмулируется программно, производительность сокращается в итоге до малой части той, которую можно было бы получить при непосредственном выполнении операционной системы на той же аппаратуре.

Проект *plex86* (*http://savannah.nongnu.org/projects/plex86*) использует еще один подход, реализуя виртуальную среду, в которой может выполняться Windows или другая операционная система (или их приложения). Программы, выполняемые на виртуальной машине, работают без снижения скорости, за исключением тех случаев, когда нужно обращаться к аппаратным устройствам. Это весьма напоминает Dosemu, но реализация более надежна и не ограничивается возможностью запуска DOS.

На момент написания этой книги все проекты, которые обсуждались в данном разделе, были довольно незрелыми и имели существенные ограничения. Грубо говоря, здесь вполне применимы выражения «как повезет» и «за что заплатишь, то и получишь».

Возможно, вам больше повезет с коммерческим продуктом, таким как VMware (*http://www.vmware.com*) или Win4Lin (*http://www.win4lin.com*). Оба основаны на реализации среды виртуальной машины (как в *plex86*), поэтому для выполнения Windows-приложений нужно установить экземпляр Windows. По крайней

мере, в отношении VMWare известно, что степень совместимости очень высока. VMWare поддерживает версии DOS/Windows от MS-DOS до .NET, включая все промежуточные. Можно даже установить несколько из известных дистрибутивов Linux, чтобы запускать на одном компьютере несколько экземпляров Linux. С разной степенью ограничений могут выполняться и другие операционные системы, в том числе FreeBSD, Netware и Solaris. Хотя при этом неизбежны некоторые накладные расходы, но современные процессоры с гигагерцевыми тактовыми частотами могут обеспечить приемлемые уровни производительности для большинства стандартных приложений, таких как офисные программы.

Win4Lin – более новая разработка, чем VMWare. Когда писалась эта книга, Windows и ее приложения быстрее выполнялись под Win4Lin, чем под VMware, но поддержка существовала только для Windows 95/98/ME, а не для Windows NT/2000/XP. Как и в отношении других проектов, описывавшихся в данном разделе, мы советуем следить за состоянием разработки этого продукта, чтобы определить, когда он созреет до такой степени, что будет способен удовлетворить ваши потребности.

Доступ к удаленному рабочему столу Windows

В этом разделе главы мы рассмотрим возможность использования системы Linux в качестве тонкого клиента для терминального сервера Microsoft Windows. При таком подходе системный администратор центрального сервера, работающего под управлением Windows, устанавливает необходимые пользователям приложения (убедившись при этом, что количества существующих лицензий вполне достаточно, чтобы организовать поддержку всех пользователей), после чего пользователи обращаются к приложениям с использованием систем Linux. Зачастую при таком подходе производительность приложений превышает производительность тех же приложений, исполняющихся на локальных компьютерах, работающих под управлением Windows!

Если вам ничего не известно о предлагаемых Microsoft службах терминалов, сведения, которые приводятся в этом разделе, могут вас приятно удивить. Службы терминалов предоставляют функциональную возможность под названием RDP (Remote Desktop Protocol, или Remote Display Protocol – протокол доступа к удаленному рабочему столу, или протокол доступа к удаленному экрану), что позволяет организовать взаимодействие со свободно распространяемым продуктом под названием *rdesktop*. Таким образом, *rdesktop* представляет собой инструментальные средства для Linux, позволяющие запускать приложения на удаленных системах Microsoft: Windows NT 4.0, Windows 2000 Server, XP Pro и Windows Server 2003.

Мало кто всерьез рассматривает возможность размещения приложений на сервере, работающем под управлением Microsoft Windows. Когда корпорация Microsoft выпустила первую жизнеспособную сетевую операционную систему (Network Operating System, NOS) Windows NT версий 3.51 и 4.0, они не обладали такими возможностями. Серверы Windows NOS традиционно использовались для запуска вспомогательных офисных приложений, таких как сервер электронной почты, различные базы данных и веб-серверы. Сторонним производителем – компанией Citrix – в составе продукта WinFrames была предложена реализация

служб терминалов – многопользовательской технологии, впервые использованной в Windows NT 3.51 и позволившей создавать несколько пользовательских сеансов в ядре NT.

Практический пример использования RDP в Linux

С целью демонстрации выгод использования Linux в качестве удаленного рабочего стола автор настроил Microsoft Terminal Services для рабочей группы, нуждавшейся в доступе к Exchange Server. В качестве основных программных продуктов использовались Windows 2003 и Outlook 2000. Это позволило нашим инженерам отвечать на запросы о встречах и использовать систему планирования производства. Кроме того, это дало возможность секретарю управлять расписанием персонала отдела.

Ранее отдел информационных технологий закупал ноутбуки для каждого инженера с единственной целью организовать с помощью Outlook возможность управления своим расписанием и осуществлять планирование. Каждый ноутбук обходился отделу приблизительно в 4000 долларов. Спустя некоторое время было принято решение переоснастить ноутбуки в коммерческом отделе и организовать сервер для рабочей группы.

В новой конфигурации сервер для рабочей группы был оснащен службами терминалов, службами лицензирования и WINS. На каждом компьютере, работающем под управлением Linux, был установлен и запущен пакет Samba, а в файле настроек Samba был активирован клиент WINS. Для обслуживания персонала отдела на сервере рабочей группы использовались локальные учетные записи.

После того как удалось организовать разрешение имен NetBIOS в IP-адреса на сервере и на рабочих станциях, в группу Remote Desktop Users были добавлены локальные пользователи. Это позволило серверу Windows распознавать хосты Linux, и наоборот. После этого пользователи могли регистрироваться на сервере и запускать приложения, способные исполняться в многопользовательской среде.

Чтобы с помощью компьютера, работающего под управлением Linux, запускать Windows, вам потребуется настроить службу терминалов Windows для запуска приложений и удаленного администрирования, а также установить и настроить *rdesktop* и популярный внешний интерфейс TSClient. Дополнительно можно добавить еще один компонент под названием Virtual Network Computing (VNC – виртуальная вычислительная сеть) и обращаться к удаленным рабочим столам Windows, Macintosh OS X и Linux, используя подход, несколько отличающийся от служб терминалов.

Для использования приложений Microsoft вам придется приобрести лицензии. Для доступа к серверу Microsoft Windows необходимо приобрести лицензию для клиентского доступа (Client Access License, CAL). Кроме того, потребуется лицензия на прикладные продукты, которые планируется использовать, например Microsoft Office.

Ныне Microsoft распространяет клиентское программное обеспечение доступа к терминальному серверу для настольных компьютеров Apple и Windows. Это позволило Citrix создать терминальные расширения и предложить программные решения для UNIX и, в конечном счете, для Linux. Однако сведения, что приводятся в этом разделе, позволят организовать прямой доступ из Linux к службам терминалов Windows на основе *rdesktop* и Samba без использования расширений Citrix.

rdesktop и TSClient

Как уже отмечалось, *rdesktop* позволяет из Linux запускать Windows-приложения, размещенные на удаленной Windows-системе. Кроме того, это дает возможность из Linux выполнять удаленное администрирование Windows через службу терминалов, что позволяет одновременно пользоваться обеими операционными системами.

Аспирант университета штата Новый Южный Уэльс, Австралия, Мэтью Чапмен (Matthew Chapman) создал свободно распространяемый программный продукт – *rdesktop*, представляющий собой клиент для Windows NT Terminal Server, Windows 2000, Windows XP Professional и Microsoft Server 2003 Terminal Services. Вам не придется приобретать у Microsoft лицензию на пользование *rdesktop* (однако необходимо будет приобрести лицензии на сами Windows-приложения). *rdesktop* можно представить себе как своего рода веб-броузер или клиент FTP, Telnet или SSH.

Взаимодействие между *redesktop* и RDP от Microsoft организовано с использованием протоколов промышленного стандарта Internet Engineering Task Force (Проблемная группа проектирования Интернета), обсуждаемого в документах RFC 905 и RFC 2126. Эти документы описывают реализацию протокола совместного использования приложений Telecommunication Standardization Sector ITU-T.128 (сектор стандартизации коммуникаций Международного телекоммуникационного союза). Мэтью выпустил *rdesktop* на условиях GNU Public License (Общественная лицензия GNU) как свободное программное обеспечение, распространяемое с открытыми исходными текстами.

TSClient – это самый популярный и удобный в работе графический интерфейс к *rdesktop*. Большинство дистрибутивов Linux устанавливают *rdesktop* и TSClient по умолчанию. Однако иногда может потребоваться явно указать необходимость установки компонентов RDP в процессе инсталляции Linux или загрузить их из репозитория.

Настройка службы терминалов Windows

Если вам не приходилось заниматься настройкой серверов Windows, тогда вам трудно будет представить, какие шаги необходимо предпринять, чтобы дать возможность удаленного использования приложений Win32, расположенных на сервере NT версии 4 или 5, из Linux. Прежде всего, вам потребуется сетевая операционная система Microsoft, где можно будет установить службы терминалов. Кроме того, потребуется настроить способ, которым терминальный сервер будет определять IP-адреса рабочих станций Linux.

На рис. 28.2 демонстрируется, как выполняется настройка в операционной системе Windows 2000 Server. Обратите внимание на большое окно с заголовком

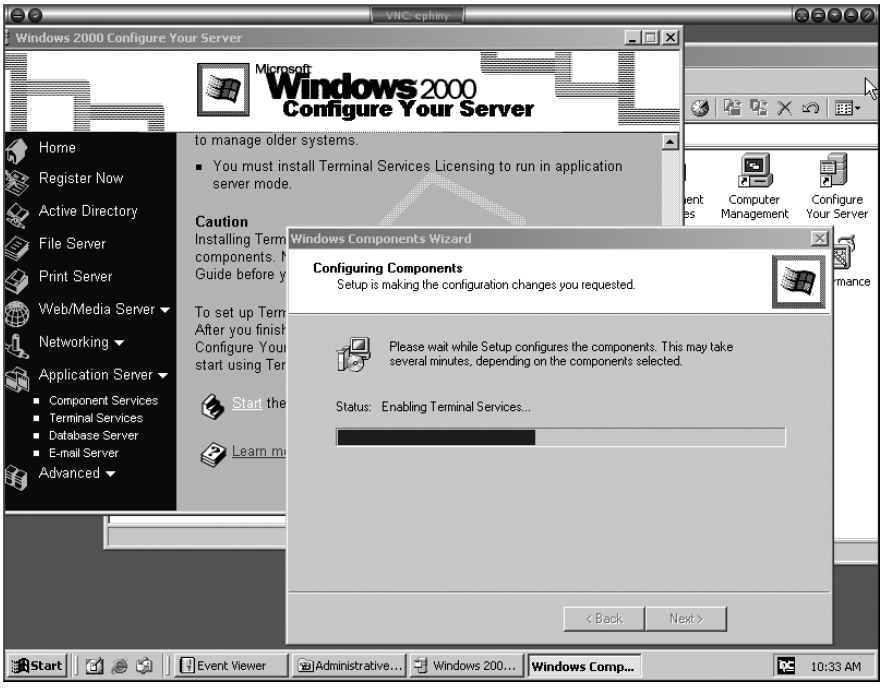


Рис. 28.2. *Настройка службы терминалов в Windows 2000 и Windows Server 2003*

Windows 2000 Configure Your Server (Настройка сервера Windows 2000). С левой стороны этого окна под заголовком Application Server (Сервер приложений) виден подзаголовок Terminal Services (Службы терминалов).

Маленькое окно на переднем плане (рис. 28.2) отображает процесс установки компонентов терминального сервера операционной системы. До появления Windows 2000 Server, чтобы иметь возможность запускать приложения из NT, необходимо было приобретать отдельный компонент. Теперь же службы терминалов являются составной частью NOS (сетевой операционной системы).

На рис. 28.3 показан тот же самый процесс в Windows Server 2003. Мастер настройки также отображается в виде диалога на экране, но он содержит более подробные инструкции по установке.

Одно из основных отличий Windows Server 2003 заключается в порядке управления лицензиями. Корпорация Microsoft требует активации лицензий клиентского доступа в последней версии сервера. Таким образом, для установки службы терминалов пользователям придется устанавливать сервер лицензий и активировать его через Интернет или по телефону.

Если у вас менее 25 пользователей, которым требуется доступ к службам терминалов, тогда служба лицензий может находиться на том же самом компьютере. Если терминальный сервер должен обслуживать более 100 пользователей, вам потребуется второй компьютер, который будет управлять службой Client Access Licenses.

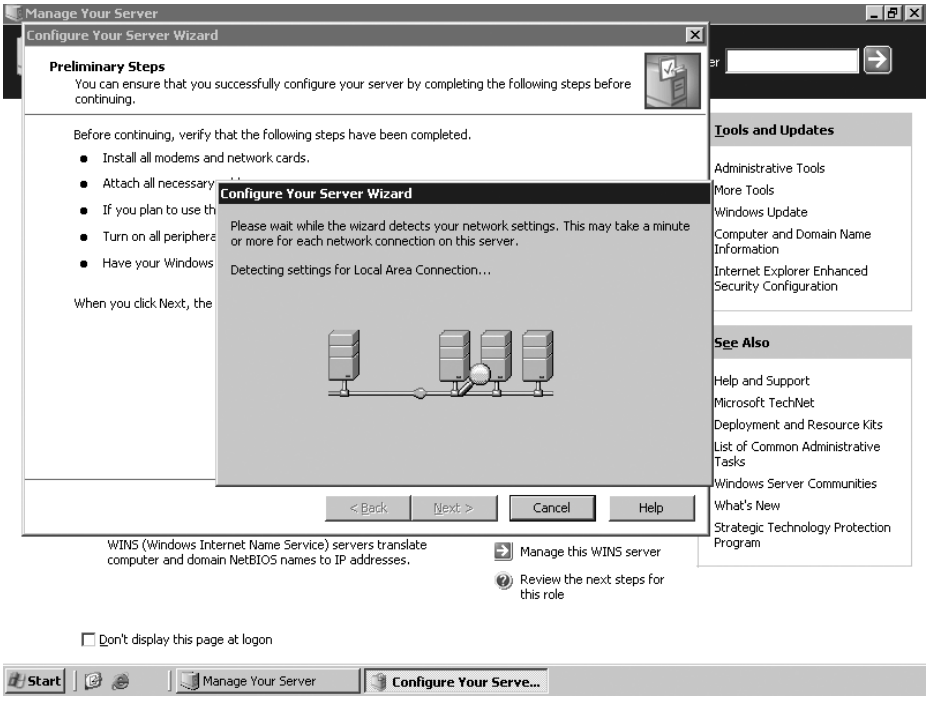


Рис. 28.3. *Мастер настройки в Windows Server 2003*

На рис. 28.4 показано, как производится настройка сервера Windows 2003 на исполнение роли терминального сервера. Для этого нужно выбрать требуемую роль и дважды щелкнуть на ней. На этом рисунке видно в выделенной строке слово Yes (Да) в колонке Configured (Настроено).

Коротко подведем итоги тому, что говорилось выше:

1. Прежде всего, необходимо установить службы терминалов на сервере Microsoft Windows, явно добавив их как компонент Windows.
2. Службы терминалов требуют наличия лицензий клиентского доступа, а ныне существующие серверы Microsoft требуют активации лицензий, в противном случае запросы на соединение будут отвергаться.
3. Для запуска приложений Microsoft в режиме службы терминалов необходимо приобретать и активировать лицензии для каждого пользователя.

Соединение с терминальным сервером

Прежде чем появится возможность запустить Windows-приложение из Linux с помощью терминального сервера, сначала необходимо настроить *rdesktop*. Самый простой способ запустить *rdesktop* заключается в использовании TSClient. Приложение TSClient – это графический интерфейс к *rdesktop*. Команду *rdesktop* можно запустить и из командной строки, но графический интерфейс TSClient упрощает создание соединений. Кроме того, он позволяет сохранять настройки под-

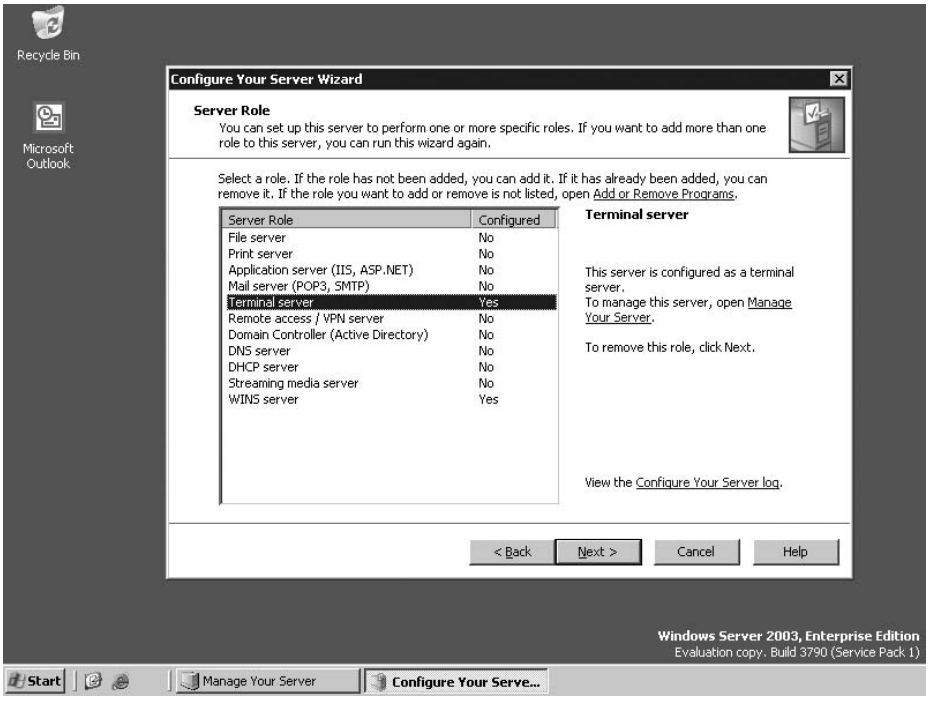


Рис. 28.4. *Параметры настройки в Windows Server 2003*

ключения к одному или более терминальным серверам. На рис. 28.5 показан TSClient, внутри которого запущено приложение Microsoft Outlook. TSClient предоставляет диалог настройки терминальных служб, аналогичный тому, что можно увидеть в Windows.

Как сделать приложения совместимыми с многопользовательской средой

В большинстве своем наиболее распространенные приложения, которые используются на терминальных серверах, – это приложения из пакета Microsoft Office. Остальные приложения, включая специализированные, требуют дополнительной модификации, чтобы иметь возможность работать в многопользовательской среде. Примеры, которые будут приведены ниже, демонстрируют порядок установки Microsoft Office.

Во время установки Office 2000 на терминальный сервер необходимо дополнительно установить Office 2000 Resource Kit и определить файл *трансформации* с расширением *.mst*. Некоторые представляют себе этот файл как файл миниспецификации. Этот файл сообщает службе терминалов, что установка компонента Office 2000 была выполнена для использования в многопользовательском режиме.

В самом начале процедуры установки необходимо указать путь к файлу трансформации из Office 2000 Resource Kit. Например, команда установки может выглядеть следующим образом:

```
E:/Set.exe G:\Program Files\ORKTools\ToolBox\Tools\Terminal Server Tools\TERMSRVR.MST
```

По умолчанию файл трансформации Microsoft SDK поставляется в составе Resource Kit. Но вы можете самостоятельно создать свой собственный файл трансформации. Некоторые приложения, такие как Internet Explorer, запускаются из меню Windows и не требуют модификаций, если вы не собираетесь менять настройки уровня безопасности. Но когда речь заходит о таких продуктах Microsoft Office, как Microsoft Project, Visio или инструменты для работы с мультимедиа, большинство администраторов стремятся ограничивать доступ к некоторым функциональным возможностям этих приложений, в зависимости от потребностей пользователей. Некоторым пользователям не нужны шаблоны создания документов, таким образом, загрузка лишних шаблонов приводит лишь к непродуктивному расходованию памяти, дискового пространства и пропускной способности сети. Отказ от использования этих возможностей может улучшить производительность.

При установке Office XP на сервер служб терминалов имеется возможность создать файл трансформации для хранения специфичных параметров настройки с помощью Office XP Custom Installation Wizard. После чего с использованием этого файла трансформации выполняется установка пакета офисных приложений на сервер служб терминалов. Все пользователи Office XP, кто зарегистрируется на сервере служб терминалов, будут получать настройки, определенные вами в файле трансформации.

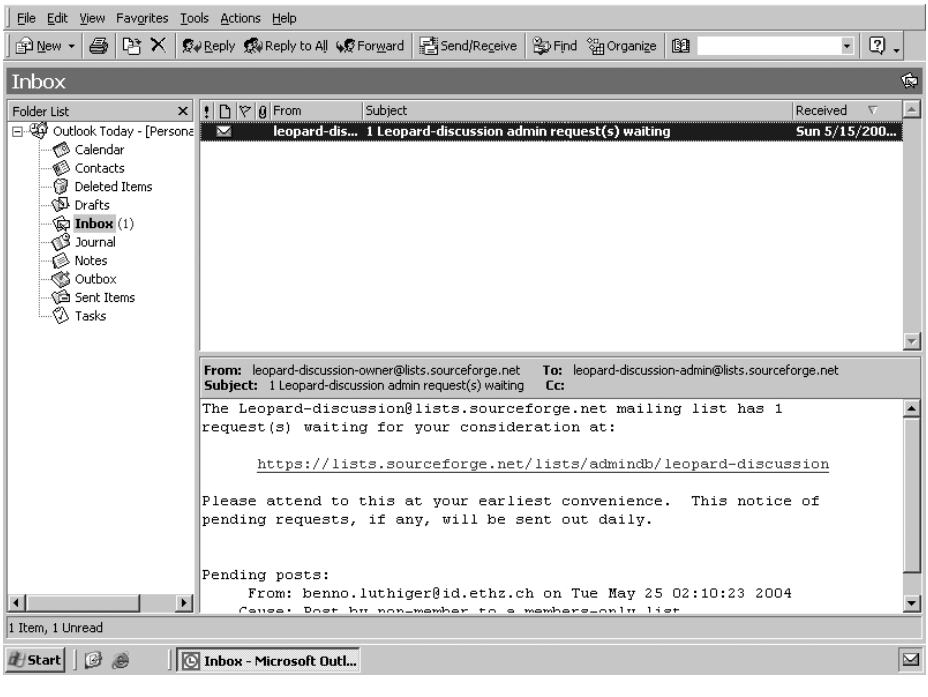


Рис. 28.5. *Microsoft Outlook в окне TSClient под управлением Linux*

Единственная версия Office 2003, которая поддерживает службы терминалов в Windows Server 2003, – это Office 2003 Enterprise Edition. Установка этого пакета выполняется проще, чем Office 2000. Программа установки автоматически распознает установку на сервер служб терминалов. По умолчанию Microsoft делает многие функциональные возможности недоступными. Пользователи Office 2003, кто зарегистрируется на сервере служб терминалов, будут получать настройки, определенные Microsoft и администратором, выполнившим установку офисного пакета.

Пользование Windows-приложениями из Linux

Чтобы запустить TSClient в дистрибутиве Fedora Core 3, достаточно выбрать пункт главного меню Launch (Загрузка)→Internet (Интернет)→Terminal Server (Терминал сервера)→Client (Клиент). После того как TSClient запустится, его необходимо настроить на работу с терминальным сервером Windows, чтобы иметь возможность запускать приложения. На рис. 28.6 показана первая группа параметров, необходимых для запуска сеанса связи.

Обратите внимание: в качестве имени сервера в первом поле мы используем имя NetBIOS. Остальные параметры определяют протокол RDP и имя домена Windows, в котором находится система. Эти сведения должны быть известны администратору, выполнявшему настройку сервера Windows, включая имя пользователя и пароль в домене Windows.

Рис. 28.6. *Клиент терминального сервера для Linux*

При желании значения параметров настройки сеанса можно сохранить в файле протокола, щелкнув на кнопке Save As (Сохранить как), благодаря чему их не придется вводить вручную в следующий раз. В данном примере мы сохранили настройки в файле *gateway.tsc*. Щелкнув на кнопке Open (Открыть), пользователь может выбрать этот файл, в результате чего все поля диалога будут заполнены сохраненными ранее значениями.

Обратите внимание: в верхней части окна клиента терминального сервера (см. рис. 28.6) имеется пять вкладок. На каждой из них находятся параметры, которые могут использоваться для настройки отображения сеанса, а также приложений, которые должны быть запущены при соединении с сервером. На рис. 28.7 продемонстрированы параметры, доступные на вкладке Display (Экран). Здесь можно определить размеры экрана и глубину цветопередачи. Уменьшая глубину цветопередачи, можно уменьшить объем трафика при работе с терминальным сервером. В данном случае мы используем глубину цветопередачи 16 бит.

На остальных вкладках содержатся дополнительные параметры, например на вкладке Local Resources (Ресурсы) можно разрешить передачу звукового потока на ваш терминал, определить комбинации клавиш, а также языковую раскладку клавиатуры. Кроме того, на следующей вкладке, Programs (Программы), можно указать, какая программа должна запускаться автоматически. Это позволяет экономить время, поскольку сразу после регистрации не нужно выбирать ярлык программы для ее запуска, если вы используете только одну программу, например Microsoft Outlook.

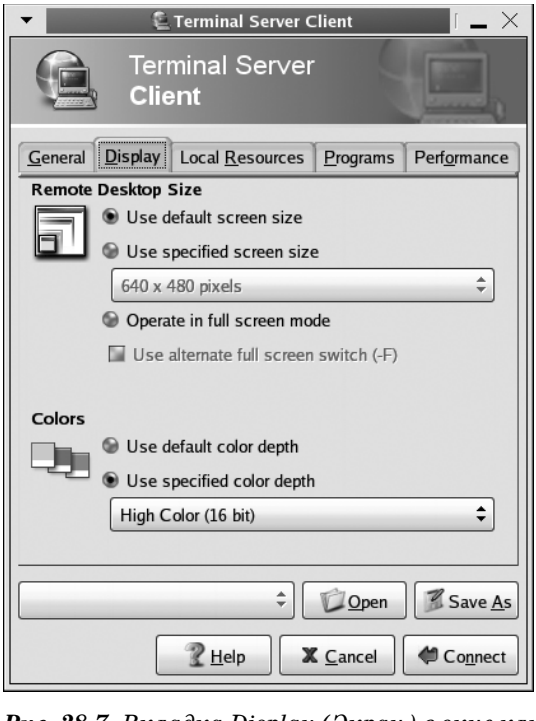


Рис. 28.7. *Вкладка Display (Экран) в окне клиента терминального сервера*

Рис. 28.8. *Экран регистрации на удаленном терминальном сервере Windows*

На вкладке Performance (Производительность) можно разрешить кэширование растрового изображения, посылку событий перемещения, привязку клавиш оконного менеджера и настройку оформления окон. При желании можно скрыть оформление окон оконного менеджера, чтобы сразу было видно, что открыт сеанс связи с терминальным сервером.

После щелчка на кнопке Connect (Соединиться) в нижней части окна программы клиента терминального сервера появится окно регистрации на сервере. Например, на рис. 28.8 показано окно регистрации для Windows Server 2003 Enterprise Edition.

Изображение удаленного рабочего стола появляется на экране в системе Linux в виде окна приложения. На рис. 28.8 на панели задач приложение с названием Gateway Terminal Service Client соответствует второй кнопке слева. После ввода имени пользователя и пароля вы будете зарегистрированы в удаленной системе, а окно приложения будет выглядеть примерно так, как показано на рис. 28.9, где показано окно приложения Outlook 2000 SR-1, запущенное в системе Windows.

Virtual Network Connection

В этом разделе мы переходим к рассмотрению широко используемой технологии доступа к удаленному столу, которая называется Virtual Network Computing (VNC – виртуальная вычислительная сеть). Мы обсудим принцип действия VNC и значимость применения этой технологии в гетерогенных сетях. Кроме того, мы рассмотрим порядок установки на различные платформы и порядок использования.

Технология VNC используется гораздо чаще, чем любые другие инструменты доступа к удаленному рабочему столу. В настоящее время существует несколько различных реализаций VNC, распространяющихся с открытыми исходными

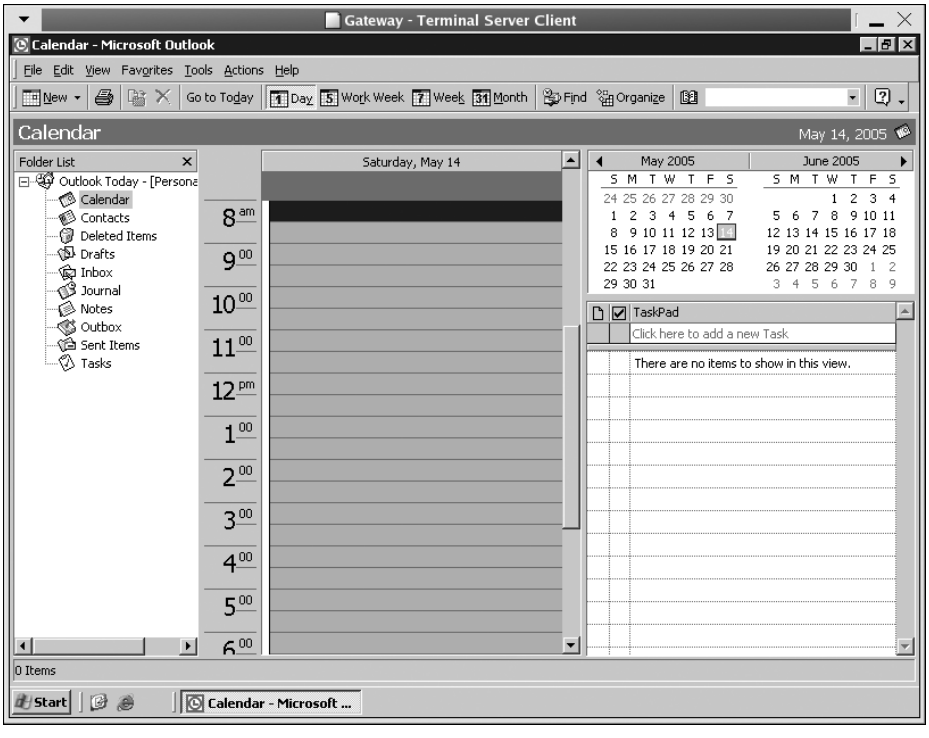


Рис. 28.9. *Microsoft Office, запущенный на терминальном сервере из Linux*

текстами. Реализации серверов существуют для Linux, Windows, Macintosh, UNIX, MS-DOS, Palm и Java. Однако не все пользователи Linux понимают ценность VNC.

Пожалуй, самое ценное в этой технологии заключается в том, что она позволяет контролировать массу компьютеров с помощью одной и той же клавиатуры, мыши и монитора. В некотором смысле VNC можно рассматривать как виртуальный мультиплексор KVM.

Мультиплексор KVM – это аппаратное устройство, позволяющее управлять несколькими компьютерами с использованием единственной клавиатуры, мыши и монитора. KVM – это акроним, составленный из начальных символов слов keyboard (клавиатура), video monitor (видеомонитор) и mouse (мышь).

На рис. 28.10 демонстрируется открытый удаленный сеанс VNC, запущенный в окружении рабочего стола GNOME, в системе Fedora Core 3. Если присмотреться внимательнее, можно заметить панели GNOME у основания экрана. При создании этого рисунка использовалась популярная реализация технологии VNC – TightVNC. На рисунке видно, что мы получили доступ к удаленному рабочему столу Windows XP и запустили приложение подключения к удаленному рабочему столу, которое можно использовать для подключения к терминальному сер-

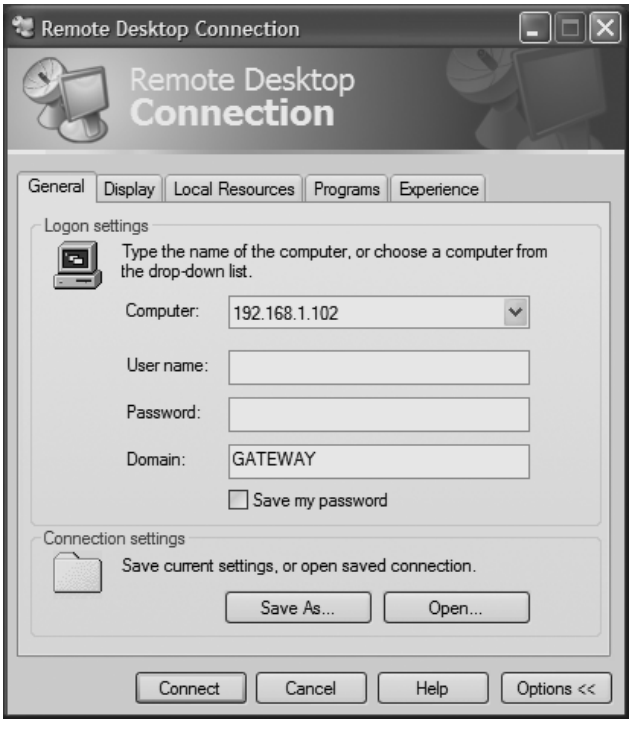


Рис. 28.10. *Запущенное приложение подключения к удаленному рабочему столу в удаленном сеансе VNC*

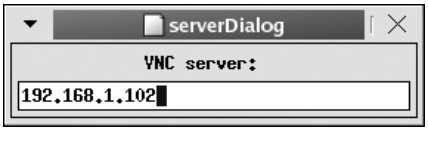


Рис. 28.11. *Диалог регистрации при подключении к удаленному рабочему столу*

веру. При использовании VNC на экране монитора отображается не только окно приложения, а удаленный рабочий стол целиком.

VNC требует, чтобы между клиентом и сервером был открыт сеанс связи. В предыдущем примере на удаленном компьютере был запущен сервер, а для открытия сеанса связи с ним была использована команда *vncviewer*. Эта команда выводит небольшое окно диалога, как показано на рис. 28.11.

На рис. 28.11 видно, насколько просто запустить удаленный сеанс:

- Запустите команду *vncviewer* из командной строки.
- Введите в диалоге IP-адрес удаленного сервера и нажмите клавишу Enter.
- В появившемся окне запроса введите пароль доступа к удаленному серверу.

После этого на экране появится окно, содержащее удаленный рабочий стол, примерно такое, как показано на рис. 28.12.

На этом рисунке демонстрируется, как выглядит удаленный рабочий стол Windows XP на экране монитора в системе Fedora Core 3 в окружении рабочего стола GNOME. На удаленном рабочем столе (см. рис. 28.12) имеют место следующие события: пользователь Linux запустил программу преобразования файлов формата PDF в формат Word, и окно подсказки приложения ZoneAlarm сообщает пользователю, что приложение пытается получить доступ к Интернету с целью поиска доступных обновлений.

VNC дает пользователю возможность взять под свой контроль удаленный компьютер. Это не сеанс связи с удаленным терминальным сервером, подобный тому, что предоставляет терминальный сервер Windows, потому что здесь вы не регистрируетесь на сервере. Вам не нужно приобретать лицензию для клиентского доступа (Client Access License, CAL). В данном случае все, что делается в рамках сеанса VNC, выполняется с разрешения пользователя, который зарегистрировался в удаленной системе.

Посмотрим, как все это работает.

Клиент VNC устанавливает соединение с удаленным компьютером посредством локальной сети или через Интернет по протоколу TCP/IP. Для организации взаимодействий VNC пользуется протоколом Remote Frame Buffer (RFB – удаленный буфер кадров). Служба RFB захватывает изображение экрана и отправляет его клиенту в ответ на запрос. Данные с изображением экрана преобразуются сервером в специальный формат, совместимый с любыми типами клиентов,

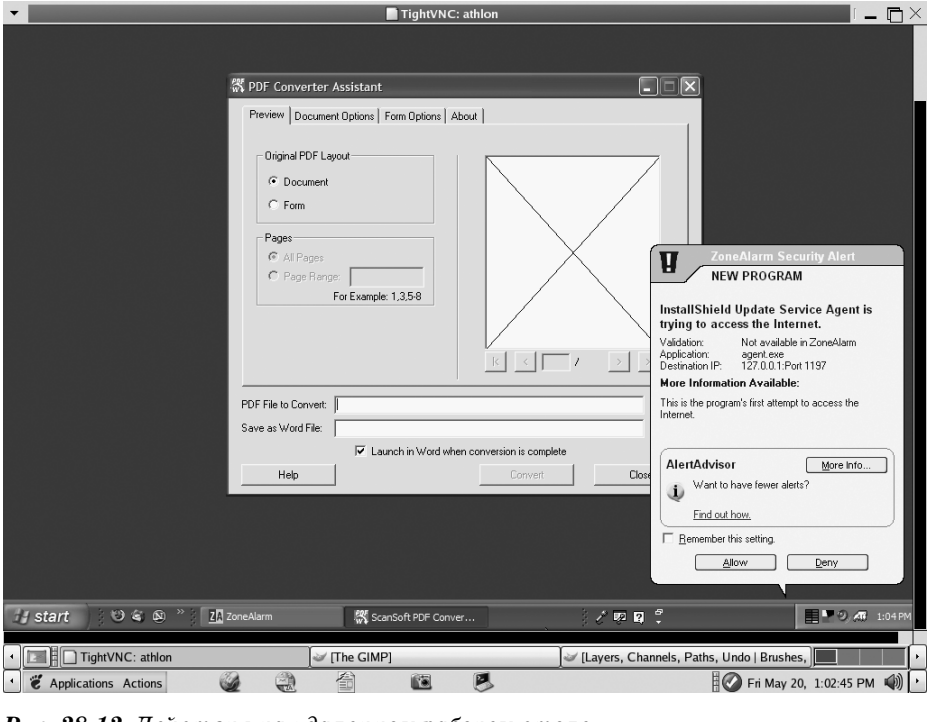


Рис. 28.12. *Действия на удаленном рабочем столе*

поддерживаемыми различными операционными системами. Клиент принимает данные от сервера и отображает изображение удаленного рабочего стола в окне на рабочем столе клиента. Все перемещения мыши и нажатия клавиш клиент отправляет серверу, предоставляя тем самым пользователю возможность управления удаленным рабочим столом.

В технологии VNC изображение экрана передается от сервера клиенту. Изображение сжимается с учетом пропускной способности сети и мощности процессоров на обоих концах соединения. Как только клиент примет изображение экрана в первый раз, сервер будет передавать только те части экрана, которые претерпели изменения. Основное содержимое кадра остается неизменным.

При использовании VNC создаются сеансы, не имеющие информации о своем состоянии. Пользователь может в любой момент прервать сеанс и подключиться к другой системе, чтобы продолжить работу с того момента, где она была прервана. Многие используют эту возможность для обеспечения мобильности.

Соединение VNC может быть инициировано как со стороны клиента, так и со стороны сервера. Обычно соединение инициируется со стороны клиента, но иногда для обеспечения технической поддержки бывает удобно использовать возможность инициации соединения со стороны сервера с клиентом, ожидающим подключения. После этого персонал отдела технической поддержки сможет показать пользователю пути решения возникшей проблемы, так же как при личной встрече, даже если их отделяют многие километры расстояния.

Основные преимущества VNC заключаются в следующем:

- Эта технология позволяет любому клиенту подключаться к серверу, независимо от типов используемых операционных систем.
- За счет шифрования обеспечивается безопасность данных, передаваемых через соединение.
- Имеются свободные реализации, распространяемые на условиях лицензии GPL.

Текущие реализации VNC (клиенты и серверы) в значительной степени основаны на протоколах TCP/IP, что позволяет пользоваться ими в самых разных сетях. Вообще есть определенный смысл создать реализацию VNC на основе какого-нибудь другого надежного двунаправленного протокола, но в настоящее время таких систем практически не существует.

Настройка VNC

Получить реализацию VNC можно из разных источников. Большинство предпочитает использовать либо RealVNC, либо TightVNC, которые можно загрузить с сайтов *http://www.realvnc.com* и *http://www.tightvnc.com*, соответственно. Загрузите программное обеспечение с одного из этих сайтов и установите его на своей платформе. Чтобы использовать VNC, необходимо запустить сервер VNC на одной системе, а затем подключиться к нему из другой системы с помощью клиента VNC.

В процессе инсталляции ПО сервера VNC для Windows (WinVNC) будет создана группа RealVNC в меню Start (Пуск). Затем, как показано на рис. 28.13, можно будет запустить сервер VNC. После того как сервер будет запущен, инициировать сеанс связи с ним можно будет, запустив команду *vncviewer* на удаленном клиенте.

Рис. 28.13. *Запуск сервера WinVNC из группы RealVNC в меню Start (Пуск)*

При первом запуске сервер VNC потребует установить пароль доступа. В Windows в системном трее появится ярлык. Щелкнув на этом ярлыке дважды, вы получите доступ к параметрам настройки сервера. При установлении соединения с удаленной машиной также потребуется ввести пароль. Для установления соединения используется достаточно безопасный протокол типа «запрос-ответ». Однако для обеспечения защищенного соединения между клиентом и сервером вам может потребоваться отыскать решение сторонних производителей.

Протокол RFB в обычном режиме не использует шифрование данных. По этой причине многие используют протокол OpenSSH или его разновидности и организуют соединение VNC через зашифрованный туннель. Описание использования VNC в паре с OpenSSH выходит за рамки данной книги. Однако в Интернете можно найти массу статей, который помогут вам создать шифрованный туннель для VNC. При поиске решений обращайте внимание на типы операционных систем, которые будут использоваться.

Запуск сервера VNC в Linux

С точки зрения приложений Linux, сервер VNC проявляет себя как дисплей X. Приложения продолжают работать на удаленной системе независимо от того, ус-

тановлено ли соединение с удаленным клиентом. Запускается сервер VNC командой:

```
$ vncserver
```

Вероятно, вам потребуется запустить сервер в фоновом режиме, чтобы он не прекратил работу по окончании вашего сеанса работы в системе.

При первом запуске сервер VNC потребует указать пароль доступа к системе. Этот пароль будут использовать те, кому потребуется соединиться с компьютером, где активизирован сервер VNC. Все серверы на одной и той же машине Linux по умолчанию будут использовать один и тот же пароль. Если позднее потребуется изменить пароль, сделать это можно будет с помощью команды:

```
$ vncpasswd
```

После запуска эта команда сначала попросит указать старый пароль, а затем новый, который будет использоваться в дальнейшем.

В типичной системе X Window имя дисплея X рабочей станции составляется из имени хоста и номера дисплея, например: `hostname:0`.

В Linux имеется возможность запустить несколько серверов VNC. Каждый из них получит имя дисплея `hostname:1`, `hostname:2` и т. д., как если бы система была оснащена несколькими мониторами. Программа *vncviewer* выбирает дисплей с первым доступным номером и сообщает об этом. В некоторых ситуациях могут существовать сеансы, которые необходимо предоставить для использования другим людям. Чтобы вынудить приложения использовать не обычный дисплей X, а уже запущенный сервер VNC, следует определить значение переменной окружения `DISPLAY` в соответствии с требуемым сервером VNC или запускать приложения с параметром *-display*, например:

```
$ xterm -display hostname:2 &
```

Для остановки требуемого сервера VNC в UNIX можно воспользоваться примерно такой командой:

```
$ vncserver -kill :2
```

FreeNX: Linux как сервер удаленного рабочего стола

Представьте себе технологию X-сервера со сжатием настолько сильным, что сеансы работы в окружении рабочего стола GNOME или KDE можно будет проводить посредством модемов с использованием шифрования SSH и впечатляюще коротким временем отклика. FreeNX – это дополнение к линейке продуктов доступа к удаленному рабочему столу, обладающее ошеломляющей производительностью. Тонкие клиенты используют незначительную долю полосы пропускания, обрабатывая обмен аудио- и видеоинформацией, обслуживая печать и другие тяжеловесные приложения, а также позволяют приостанавливать сеансы связи вместо их завершения. Пока вы предпочитаете использовать в первую очередь Linux, FreeNX обеспечит по-настоящему виртуальные мультиплексоры KVM без аппаратного обеспечения.

FreeNX отличается от Windows RDP и VNC тем, что превращает операционную систему Linux в источник приложений, используемых людьми. Так, если потребуется настроить сервер Linux и минимальными аппаратными средствами обеспечить доступ удаленным пользователям к приложениям из пакета OpenOffice.org или веб-броузеру Firefox, лучшим выбором будет FreeNX. Кроме того, при наличии клиентов, работающих с такими операционными системами, как Windows 98 или Macintosh OS X, имеется возможность получить свободно распространяемое клиентское программное обеспечение на сайте *http://nomachine.com*, которое позволит подключаться к серверу Linux из этих систем и запускать необходимые приложения.

При работе с сервером FreeNX в Linux для распределенных вычислений создается защищенная среда. Конечно, клиенты могут использовать операционную систему Linux, но FreeNX в состоянии создавать сеансы связи с клиентами, работающими в самых разных операционных системах, таких как Windows и Macintosh, и при этом не требуют наличия X Window на этих компьютерах. Помимо упоминавшихся выше, существуют реализации клиентов для PlayStation2, iPAQ и Zaurus 5XXX.

Системные администраторы предпочитают использовать FreeNX, потому что с помощью этого продукта они могут контролировать доступность функциональных возможностей и данных для пользователей. Кроме того, они могут контролировать каждый сервер Linux, находящийся в вычислительном центре, с помощью единственной клавиатуры, мыши и монитора. При этом нет необходимости приобретать и устанавливать дополнительное аппаратное обеспечение, которое позволяло бы переключать кабели клавиатуры, мыши и монитора. Они могут также отобразить на одном рабочем столе несколько окон и одновременно контролировать работу множества серверов, чего нельзя сделать с помощью мультиплексора KVM, потому что он позволяет подключаться только к одному серверу в один и тот же момент времени.

Авторство изобретения NX принадлежит Жану Филиппо Пинцари (Gian Filippo Pinzari). За основу он взял небезопасный протокол взаимодействия X-клиентов и X-сервера и довел его сжатие до такой степени, что стало возможным использовать протокол на каналах с очень невысокой пропускной способностью. Его компания, NoMachine.com, выпустила программный код в 2003 году на условиях лицензии GPL.

Теперь рассмотрим порядок настройки и использования FreeNX. Для нашего примера мы будем использовать два свободно распространяемых дистрибутива Linux – Fedora и Ubuntu. Сначала установим программное обеспечение FreeNX на Ubuntu, после того как оно будет получено в сообществе Ubuntu, на сайте *http://backports.ubuntuforums.org*. Следуя указаниям на сайте, добавьте рекомендованные зеркала в файл */etc/apt/sources.list*. Затем командой *apt-get install FreeNX* выполните установку FreeNX на сервер.

После установки добавьте пользователя, как показано на рис. 28.14.

Затем нужно выйти из системы и вновь зарегистрироваться в ней, после чего в меню Applications появится новый пункт, как показано на рис. 28.15.

Далее с сайта *http://fedoranews.org/contributors/rick_stout/freenx* следует получить и установить пакет RPM для Fedora. Убедитесь, что установлены оба компонента – клиент и сервер. И снова необходимо добавить пользователя.

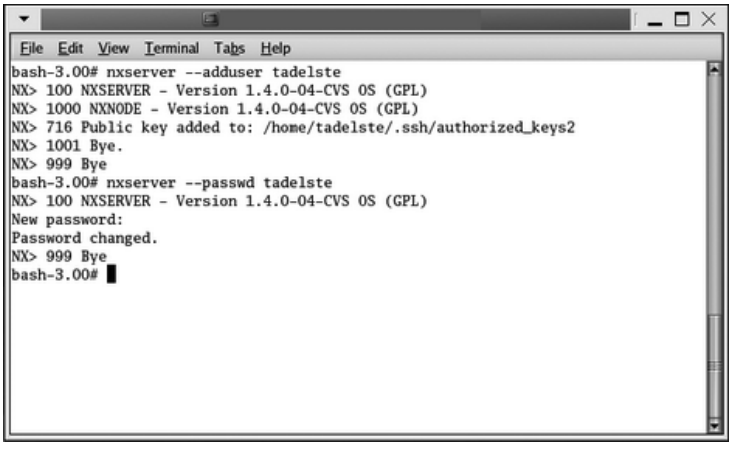


Рис. 28.14. *Добавление пользователя сервера FreeNX*

Рис. 28.15. *Пункт запуска FreeNX в меню рабочего стола*

Настройка клиента осуществляется с помощью мастера. Мастер сообщит: «The most important part of the initial connection is the key file. This file, *client.id_dsa.key*, must be copied from the server to your client machine» (Самая важная деталь, которая необходима для соединения, – это файл ключа. Данный

файл, *client.id_dsa.key*, следует скопировать с сервера на машину клиента). Следуя этому указанию, я запустил команду:

```
bash-3.00# scp /var/lib/nxserver/home/.ssh/client.id_dsa.key
username@192.168.1.109:~/
The authenticity of host '192.168.1.109 (192.168.1.109)' can't be
established.
RSA key fingerprint is 40:54:e3:c9:5e:81:39:2d:ac:70:b9:bf:44:a9:ec:a8.
Are you sure you want to continue connecting (yes/no)? yes
Warning: Permanently added '192.168.1.109' (RSA) to the list of known
hosts.
Password: здесь был введен пароль
client.id_dsa.key                             100%  672     0.7KB/s
00:00
bash-3.00#
```

После выполнения этих действий должна появиться возможность использовать сервер FreeNX для подключений с удаленными клиентами. Если вам придется столкнуться с какими-нибудь проблемами, в Интернете вы найдете множество сайтов, которые помогут найти решение. Кроме того, ответы на некоторые вопросы можно найти в списке рассылки FreeNX или в его архиве на сайте *http://developer.berlios.de/projects/freenx*.

Сеанс FreeNX запускается очень быстро. Вы можете также заметить, что FreeNX в состоянии приостанавливать сеансы, не закрывая их. Когда возобновляется приостановленная сессия, клиент повторно выполняет вход в систему, но при этом приостановленный ранее сеанс продолжается с того места, где он был прерван. Хотя, строго говоря, эта разновидность сеансов по-прежнему не имеет информации о своем состоянии, тем не менее они экономят существенную часть пропускной способности (рис. 28.16 и 28.17).

FreeNX обладает массой преимуществ для пользователей Linux. Данное программное обеспечение предоставляет превосходную среду окружения тонкого клиента, высокую скорость работы и использует надежное шифрование данных на основе продукта OpenSSH, который соответствует главным тестовым критериям стандарта FIPS 140 в форме исходных текстов. Кроме того, программное обеспечение FreeNX доступно для большинства коммерческих дистрибутивов Linux. Свободно распространяемые дистрибутивы, такие как Fedora Project и Ubuntu, предоставляют широчайшую поддержку в своих сообществах.

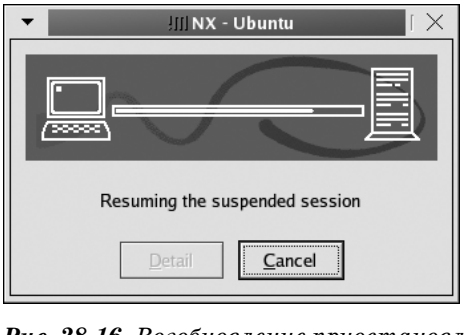


Рис. 28.16. *Возобновление приостановленного сеанса с Fedora на Ubuntu*

Рис. 28.17. *Сеанс FreeNX, запущенный на Ubuntu с сервером на Fedora Core 3*

Наконец, FreeNX может использовать сервер Linux в качестве прокси-сервера для организации доступа к серверам VNC и RDP. Для этого можно запустить *vncviewer* или *rdesktop* на сервере Linux и использовать эти удаленные приложения для запуска сеанса с операционной системой Windows. При использовании FreeNX повышается производительность сеансов VNC, выполняется шифрование данных и обеспечивается более широкий доступ к приложениям Windows через RDP.

A

Источники информации по Linux

В этом приложении перечислены различные сетевые ресурсы информации по Linux. Все эти документы доступны в электронном виде через Интернет, но многие из них имеются в печатной форме.

Дистрибутивы Linux часто включают в себя некоторый объем документации подобного рода и обеспечивают ее доступность в работающей системе. Как уже отмечалось ранее, документацию в системе Linux можно найти в самом разном представлении, например в виде страниц справочного руководства UNIX, страниц GNU Info и справочной документации в формате HTML (документы в таком представлении, например, выводит центр справки KDE).

В большинстве дистрибутивов Linux документация по отдельным программам, такая как файлы *README* и примечания к выпуску, хранится в каталоге `/usr/share/doc`. Если в системе были установлены исходные тексты ядра, поставляемую с ними документацию, как правило, можно найти в каталоге `/usr/src/linux/Documentation`.

Информацию более интерактивного характера пользователи Linux обычно получают из следующих источников:

Телеконференции

Большинство телеконференций, относящихся к Linux, находится в иерархии *comp.os.linux*, но существует немало более узконаправленных телеконференций, относящихся к конкретному дистрибутиву или посвященных проектам с открытыми исходными текстами.

IRC

Internet Relay Chat (IRC) – традиционная чат-система UNIX, которую часто используют для получения немедленных ответов на вопросы, задаваемые пользователями.

Почтовые списки рассылки

Большинство проектов Linux и open source, от ядра до KDE, пользуется почтовыми списками рассылки в качестве главного средства связи между разработчиками проекта. Во многих группах пользователей существуют свои почтовые списки с местным уклоном.

Проект документирования Linux (LDP)

Основным источником свободно распространяемой документации по Linux служит Проект документирования Linux (Linux Documentation Project, LDP). Главный веб-сайт LDP находится по адресу *http://www.tldp.org*, но кроме него существует масса зеркальных сайтов, раскиданных по всему свету, и какой-нибудь из них может оказаться ближе к вам или быть менее загруженным.

Документация в LDP организована по нескольким типам. *Guide* – это длинные, часто объемом с книгу, руководства, детально описывающие такие большие темы, как, скажем, работу в сети. HOWTO – документы среднего объема, описывающие решение конкретных задач, как, например настройку звуковых карт. Описания решений более мелких задач по специальным темам, для которых нет смысла создавать полноценный документ HOWTO, можно найти в mini-HOWTO. Наконец, существует целый ряд сборников FAQ с ответами на часто возникающие вопросы по Linux.

Документы LDP предоставляются в ряде различных форматов, в том числе HTML, простой текст, PDF и PostScript. Многие документы переведены на другие языки группой добровольцев.

FTP-сайты

Хотя в вашем дистрибутиве Linux пакеты многих приложений уже находятся в скомпилированном виде, нередко возникает необходимость компилировать приложения из исходного программного кода, потому что приложение недоступно в скомпилированном виде, либо вы хотите посмотреть на исходный программный код, либо просто предпочитаете самостоятельно собирать приложения из исходных текстов. Ниже приводится несколько популярных сайтов, где можно взять исходные тексты программ для Linux.

Многие из этих сайтов пользуются исключительной популярностью и практически всегда перегружены. По этой причине мы настоятельно рекомендуем вам пользоваться зеркальными сайтами (они регулярно загружают программное обеспечение с главного сайта), которые географически расположены к вам ближе. Обычно с зеркальным сайтом легче соединиться, и работает он быстрее.

Помимо перечисленных здесь FTP-сайтов, многие веб-сайты, упоминаемые в этом приложении, имеют соответствующие FTP-сайты, откуда можно загрузить необходимые файлы.

FTP-сайт	Описание
ftp://ftp.gnu.org	Основной сайт для загрузки проекта GNU
ftp://ftp.ibiblio.org	Огромный архивный сайт Linux и один из первых архивных сайтов Linux (ранее известный под названием *sunsite.unc.edu*)
ftp://ftp.x.org	Архив программного обеспечения для X Window System

Сайты World Wide Web

В этом разделе перечислены лишь некоторые из тысяч веб-сайтов Интернета, посвященных операционной системе Linux, с разделением на категории, выбранные достаточно условно. Из-за динамической природы Web некоторые из них уже не будут действовать и, безусловно, появится много новых, когда вы будете читать эти строки.

Общая документация

Эти сайты предлагают электронную документацию, статьи о Linux или информацию, относящуюся к отдельным областям Linux.

Веб-сайт	Описание
http://www.andamooka.org	Данный веб-сайт содержит огромное число книг, часть из которых имеют отношение к операционной системе Linux, например «KDE 2.0 Development»
http://www.justlinux.com	Сайт с новостями и форумами
http://www.linas.org/linux	Сайт компании Linux Enterprise Computing
http://www.linux-laptop.net	Сайт, посвященный вопросам использования Linux на портативных компьютерах – ноутбуках
http://www.linuxfocus.org	Свободно распространяемый электронный журнал «Linux Focus»
http://www.linuxgazette.com	«Linux Gazette» – свободно распространяемый ежемесячный электронный журнал
http://www.linuxjournal.com	Веб-сайт журнала «Linux Journal»
http://www.linuxmagazine.com	Веб-сайт журнала «Linux Magazine»
http://www.linuxquestions.org	Очень популярный и весьма содержательный сайт Linux Questions
http://www.tldp.org	Основной сайт проекта Linux Documentation Project

Проекты open source

Здесь перечислены веб-сайты некоторых наиболее популярных проектов open source и бесплатного программного обеспечения.

Веб-сайт	Описание
http://freedesktop.org	Проект Freedesktop.org поддерживается несколькими крупными поставщиками программного обеспечения с целью согласованного развития проектов рабочих столов и обеспечения единства функциональных возможностей
http://koffice.kde.org	Сайт проекта разработки пакета офисных приложений KDE Office Suite
http://www.abisource.com	Текстовый процессор AbiWord

Веб-сайт	Описание
http://www.alsa-project.org	Проект альтернативной звуковой системы для Linux (ALSA – Alternative Linux Sound Architecture)
http://www.apache.org	Сайт проекта разработки веб-сервера Apache
http://www.cups.org	Сайт проекта универсальной системы печати для UNIX (CUPS – Common UNIX Printing System)
http://www.gnome.org	Сайт проекта GNOME Desktop
http://www.gnu.org	Сайт проекта GNU Project
http://www.isdn4linux.de	Сайт проекта ISDN4Linux, посвященный поддержке ISDN в Linux
http://www.kde.org	Сайт проекта K Desktop Environment (KDE)
http://www.kernel.org	Официальный сайт ядра Linux
http://www.linux-usb.org	Сайт проекта Linux USB
http://www.mozilla.org	Сайт проекта по разработке веб-броузера Mozilla
http://www.mysql.com	Сайт компании MySQL AB, ведущей разработку системы управления базами данных MySQL
http://www.openoffice.org	Сайт проекта разработки пакета офисных приложений OpenOffice.org – свободно распространяемой версии пакета StarOffice
http://www.postfix.org	Проект почтовой программы Postfix
http://www.povray.org	Пакет трехмерной векторной графики The Persistence Of Vision Raytracer
http://www.winehq.com	Сайт проекта Wine
http://x.org	Сайт поддержки проекта X Window System

Языки программирования и инструментальные средства

Ниже перечислены сайты, имеющие отношение к языкам программирования, популярным в Linux, и предоставляющие хостинг для многих программных проектов.

Веб-сайт	Описание
http://www.source-forge.net	Головной сайт огромного числа проектов по разработке программного обеспечения для операционной системы Linux. Предоставляет место для размещения документации, репозиториев с исходными текстами, сведений об обнаруженных ошибках и сборки приложений
http://savannah.gnu.org	Сайт GNU Savannah предлагает услуги, аналогичные SourceForge, но имеет официальное признание Фонда свободного программного обеспечения, поскольку все программное обеспечение сайта лицензировано по GPL
http://www.blackdown.org	Домашняя страница проекта переноса Java на платформу Linux

Веб-сайт	Описание
http://www.perl.com	Официальный сайт языка программирования Perl
http://www.php.net	Веб-сайт языка программирования PHP
http://www.python.org	Домашняя страница языка программирования Python

Новостные и информационные сайты

На этих сайтах предлагаются новости, которые будут интересны пользователям Linux.

Веб-сайт	Описание
http://www.desktoplinux.com	Сайт посвящен вопросам использования Linux в качестве системы рабочего стола
http://www.linux.com	Сайт (с очень хорошим названием) содержит сведения общего характера и новости о Linux
http://www.linuxtoday.com	Веб-сайт Linux Today
http://www.lwn.net	Веб-сайт Linux Weekly News, где очень подробно освещаются темы, связанные с разработкой ядра операционной системы
http://linuxsecurity.com	Новости и информация о проблемах безопасности Linux
http://www.newsforge.com	Веб-сайт NewsForge
http://www.slashdot.com	Популярный новостной сайт и дискуссионный клуб, который заявляет о себе следующим образом: «Новости для компьютерных фанатов. Все самое главное»
http://www.theregister.co.uk	Британский веб-сайт The Register, где размещаются новости индустрии информационных технологий, имеющие отношение к Linux
http://www.varlinux.org	Новостной сайт VarLinux для Value Added Resellers (VAR)

Сайты каталогов и загрузки программного обеспечения Linux

Ниже перечислены несколько сайтов, где хранятся большие библиотеки программного обеспечения Linux с возможностью поиска, и где можно найти ссылки на другие сайты загрузки.

Веб-сайт	Описание
http://www.freshmeat.net	Огромный каталог программного обеспечения open source для Linux
http://www.icewalkers.com	Еще один большой каталог программного обеспечения для Linux
http://www.linuxberg.com	Каталог программного обеспечения

Дистрибутивы Linux

Ниже приводится достаточно длинный, но отнюдь не исчерпывающий список различных дистрибутивов Linux. Сюда входят и дистрибутивы, созданные большими компаниями, такими как Red Hat, и специализированные дистрибутивы, разработанные отдельными людьми или небольшими группами. Сайт DistroWatch (*http://distrowatch.com*) предоставляет интересные новости и статистическую информацию о большом числе существующих дистрибутивов.

Веб-сайт	Описание
http://www.debian.org	Debian GNU/Linux. Популярный дистрибутив, разрабатываемый сообществом добровольцев
http://fedora.redhat.com	Fedora Core – быстро развивающаяся, свободно распространяемая версия Red Hat
http://www.gentoo.org	Gentoo Linux – стремительно развивающийся дистрибутив, разрабатываемый сообществом добровольцев. Отличительной особенностью этого дистрибутива является его ориентированность на распространение в виде исходных текстов
http://www.knoppix.net	Knoppix. Дистрибутив, работающий с компакт-диска, прекрасно подходит для первого знакомства с операционной системой Linux и для решения задач по восстановлению системы после фатальных сбоев
http://www.kubuntu.org.uk	Kubuntu Linux. Версия Ubuntu, которая предлагает KDE в качестве рабочего стола
http://linspire.com	Linspire. Коммерческая версия настольной системы для конечных пользователей
http://www.lycoris.com	Lycoris
http://www.mandriva.com	Mandriva Linux
http://www.opensuse.org	OpenSuSE – свободно распространяемая версия SUSE Linux
http://www.redhat.com	Red Hat Linux
http://www.slackware.com	Slackware Linux
http://www.suse.com	SUSE Linux, ныне распространяется компанией Novell
http://www.turbolinux.com	Turbolinux. Этот дистрибутив пользуется популярностью в Восточной Азии
http://www.ubuntulinux.org	Ubuntu Linux. Дистрибутив настольной системы для конечных пользователей
http://www.xandros.com	Xandros Desktop Linux
http://www.yellowdoglinux.com	Yellow Dog Linux. Этот дистрибутив предназначен для работы на аппаратной платформе Macintosh

Компании, занимающиеся коммерческой разработкой программного обеспечения для Linux

Перечисленные ниже компании предлагают программное обеспечение для платформы Linux и услуги на коммерческой основе.

Веб-сайт	Описание
http://www.codeweavers.com	CodeWeavers, разработчик CrossOver Office и CrossOver Plugin – продуктов, основанных на программном обеспечении проекта Wine, предлагающих возможность запуска приложений Windows под управлением ОС Linux.
http://www.trolltech.com	TrollTech, разработчик Qt – платформонезависимой библиотеки разработки графического интерфейса. Qt лежит в основе KDE.
http://www.vistasource.com	VistaSource, ранее эта компания называлась Applix, разработчик Applixware Office Suite.
http://www.vmware.com	VmWare распространяет программное обеспечение, реализующее виртуальные машины, которое позволяет запускать операционные системы одну в другой, например Windows в Linux, и наоборот.

Стандарты Интернета и другие стандарты

Ниже приводится список сайтов, где можно найти описания стандартов, лежащих в основе Linux и Интернета.

Веб-сайт	Описание
http://www.faqs.org/rfcs	Requests For Comments (RFC) – технические документы, описывающие многие из протоколов, на которых основывается Интернет. Содержит множество других документов стандартов и FAQ.
http://www.freestandards.org	The Free Standards Group – некоммерческая организация, занимающаяся распространением технологий open source путем разработки, применения и продвижения стандартов.
http://www.linuxbase.org	The Linux Standard Base – проект Free Standards Group, созданный для разработки и распространения ряда стандартов, которые должны улучшить совместимость дистрибутивов Linux и позволить приложениям выполняться на любой совместимой со стандартами Linux-системе.
http://www.w3c.org	World Wide Web Consortium – организация, занимающаяся разработкой спецификаций, руководств, программного обеспечения и инструментальных средств для Всемирной паутины.

Разное

В заключение приводится перечень из нескольких сайтов, которые не удалось отнести к какой-либо из упомянутых выше категорий.

Веб-сайт	Описание
http://counter.li.org	Linux Counter – уникальный сайт, где собрана информация, позволяющая оценить число пользователей Linux во всем мире.
http://www.li.org	Linux International – некоммерческая организация, занимающаяся пропагандой Linux и Linux-сообщества.

Алфавитный указатель

Символы

Числа

A

B

C

D

E

F

G

H

I

J

K

L

M

N

O

P

Q

R

S

T

U

V

W

X

Y

Z

А

Б

В

Г

Д

Ж

З

И

K

Л

М

H

О

П

P

C

T

У

Ф

X

Ц

Ч

Ш

Э

Я