Михаэль Кофлер

Linux

Установка, настройка администрирование

Addison-Wesley

ПИТЕР

Michael Kofler

Linux 2013

Das Desktop- und Server-Handbuch für Ubuntu, Debian, CentOS und Co.

12., überarbeitete und erweiterte Auflage

An imprint of Pearson

München ● Boston ● San Francisco ● Harlow, England
Don Mills, Ontario ● Sydney ● Mexico City
Madrid ● Amsterdam

Михаэль Кофлер

Linux

Установка, настройка администрирование

ПИТЕР®

Москва • Санкт-Петербург • Нижний Новгород • Воронеж
Ростов-на-Дону • Екатеринбург • Самара • Новосибирск
Киев • Харьков • Минск
2014

М. Кофлер

Linux. Установка, настройка, администрирование

Перевел с немецкого О. Сивченко

Заведующий редакцией	*Д. Виницкий*
Ведущий редактор	*Е. Каляева*
Научный редактор	*О. Сивченко*
Художник	*Л. Адуевская*
Корректоры	*О. Андриевич*
Верстка	*М. Моисеева*

ББК 32.973.2-018.2
УДК 004.451

Кофлер М.

К74 Linux. Установка, настройка, администрирование. — СПб.: Питер, 2014. — 768 с.: ил.

ISBN 978-5-496-00862-4

Linux, наряду с Microsoft Windows и Apple OS X, является одной из важнейших операционных систем для ПК. Системы Linux установлены на многих крупных серверах и применяются на многочисленных смартфонах и планшетах, работающих на базе Android.

Раньше существовали предрассудки о том, что Linux, якобы, слишком сложна и неудобна в использовании, но сегодня они преодолены. Конечно, Linux работает не совсем так, как Windows, но перейти с Windows на Linux не сложнее, чем с Windows 7 на Windows 8. При этом большинство дистрибутивов Linux бесплатны, а также значительно более надежны, чем Windows. В Linux также не заводятся вирусы и трояны.

Но если Linux так проста в использовании, зачем было писать о ней такую толстую книгу? На это есть несколько причин.

Linux — это не просто альтернативная операционная система, способная заменить Windows на ПК. В Linux есть многочисленные дополнительные возможности применения и функции. Речь может идти как об автоматизации повседневных задач, так и о многом другом, вплоть до конфигурации сетевых серверов. Linux включает в себя множество различных дистрибутивов. Наиболее известными считаются Debian, Red Hat, openSUSE и Ubuntu. Автор выстроил эту книгу в универсальном ключе, без привязки к конкретным дистрибутивам. Издание призвано научить читателя не только работать с Linux, но и понимать ее. Давайте научимся все делать по-линуксовски!

12+ (В соответствии с Федеральным законом от 29 декабря 2010 г. № 436-ФЗ.)

ISBN 978-3827332080 нем.
ISBN 978-5-496-00862-4

© 2013 by Addison-Wesley Verlag
© Перевод на русский язык ООО Издательство «Питер», 2014
© Издание на русском языке, оформление ООО Издательство «Питер», 2014

Права на издание получены по соглашению с Addison-Wesley Longman. Все права защищены. Никакая часть данной книги не может быть воспроизведена в какой бы то ни было форме без письменного разрешения владельцев авторских прав.

Информация, содержащаяся в данной книге, получена из источников, рассматриваемых издательством как надежные. Тем не менее, имея в виду возможные человеческие или технические ошибки, издательство не может гарантировать абсолютную точность и полноту приводимых сведений и не несет ответственности за возможные ошибки, связанные с использованием книги.

В оформлении обложки использованы иллюстрации shutterstock.com.

ООО «Питер Пресс», 192102, Санкт-Петербург, ул Андреевская (д Волкова), 3, литер А, пом 7Н

Налоговая льгота — общероссийский классификатор продукции ОК 005-93, том 2; 95 3005 — литература учебная

Подписано в печать 20 09 13 Формат 70×100/16 Усл п л 61,920 Тираж 2000 Заказ 0000

Отпечатано в полном соответствии с качеством предоставленных издательством материалов
в ГППО «Псковская областная типография» 180004, Псков, ул Ротная, 34

Эта книга посвящена моей жене Хайди
и моим детям Себастиану и Матиасу.

Краткое содержание

Оглавление

Предисловие

В предыдущих изданиях этой книги мне еще приходилось объяснять, что такое Linux. Эти времена уже в прошлом. Система Linux, наряду с Windows и Mac OS X, является одной из важнейших операционных систем для персональных компьютеров. Сейчас Linux владеет значительной долей серверного рынка (в частности, серверы Linux используют Google и Amazon), а также приобретает популярность как операционная система для локальных компьютеров и работает во многих встроенных системах (так называются готовые устройства, которые по внешнему виду отличаются от компьютеров). Вполне вероятно, что ваш ADSL- или WLAN-маршрутизатор также работает под управлением Linux.

Ранее существовали предрассудки о том, что система Linux неудобна в применении, но теперь пользователи так не думают. Бесспорно, работа с Linux построена иначе, чем с Windows, но переход дается не сложнее, чем с Windows XP на Windows 7. При этом большинство дистрибутивов Linux бесплатные и, в то же время, гораздо надежнее Windows. В Linux у вас не возникнет никаких проблем с вирусами или троянскими конями.

Об этой книге

Если система Linux так проста в использовании, почему же в этой книге так много страниц? На то есть несколько причин.

- Linux — это гораздо больше, чем приложение к Windows. В ней предоставляются бесчисленные дополнительные функции и возможности применения, касающиеся и автоматизации повседневных задач, и конфигурации сетевых серверов, и многого другого. И хотя сегодня почти все знакомы с браузерами, почтовыми и офисными программами, есть другие темы, более сложные в техническом отношении, которые требуют подробного описания.
- Linux — это не одна система, а набор многочисленных дистрибутивов (можно сказать проще: дистрибутив — это набор программ, работающих под Linux; наиболее известные дистрибутивы — Debian, Red Hat, openSUSE и Ubuntu). Благодаря такому разнообразию возникает масса преимуществ, но есть и один серьезный недостаток: многие детали различных дистрибутивов выполнены по-разному. В этой книге, насколько это возможно, я старался излагать материал без привязки к конкретному дистрибутиву. Но при этом все время приходилось ссы-

латься на различные варианты по следующему принципу: в Debian это работает так, в openSUSE — иначе.

- Наконец, мне хотелось, чтобы вы не просто научились работать с Linux, но и поняли данную систему. В этом отношении книга, возможно, немного неудобна: вы не найдете здесь множества скриншотов с указаниями типа «нажмите здесь». Я, напротив, хотел объяснить основы системы и немного посвятить вас в философию UNIX/Linux.

В добрый путь!

Разумеется, вы можете просто пользоваться Linux как самой обычной операционной системой для ПК. Но, в отличие от платных продуктов, Linux также позволяет практически безгранично настраивать операционную систему и приспосабливать ее под личные нужды: для программирования, работы в сети или использования компьютера в качестве сервера. Для решения практически любой задачи система предоставляет массу инструментов. И чем больше вы будете осваиваться в мире Linux, тем более удобной для вас будет эта операционная система. Желаю вам удачи в ваших экспериментах и работе с Linux!

Михаэль Кофлер
www.kofler.cc

От издательства

Ваши замечания, предложения и вопросы отправляйте по адресу электронной почты vinitski@minsk.piter.com (издательство «Питер», компьютерная редакция).

Мы будем рады узнать ваше мнение!

На сайте издательства http://www.piter.com вы найдете подробную информацию о наших книгах.

1 Быстрое введение в Linux

В этой главе вы научитесь азам работы с Linux: узнаете, как войти в систему, выполнять программы, получать доступ к файлам и носителям данных, выходить из системы или перезагружать компьютер и т. д. В данной главе сообщаются минимальные базовые сведения об управлении файлами в Linux, а также описывается, где можно найти электронную документацию в установленной системе или в Интернете.

Основная проблема, возникающая при попытке дать общую характеристику Linux, заключается в том, что практически все функции системы являются настраиваемыми. По этой причине, например, вид стартового меню Рабочего стола (аналогичное меню Пуск в Windows) в каждом дистрибутиве немного различается. Не исключено, что некоторые операции, осуществляемые с помощью мыши или горячих клавиш, будут давать в Red Hat иной результат, нежели в SUSE. Поэтому в данной главе нередко делаются оговорки *«в большинстве случаев»*, *«обычно»* и т. д. С этим ничего не поделаешь. Единственная альтернатива в данном случае — описание строго определенного дистрибутива Linux, причем с точным указанием номера версии этого дистрибутива.

1.1. Запуск и завершение работы Linux

Чтобы приступить к работе с Linux, следует перезапустить компьютер. При запуске вы указываете в специальном меню, что хотите работать с Linux, а не с Windows. Запуск Linux из Windows невозможен — за исключением случаев, когда в Windows задействуется виртуальная машина, например VMware — и Linux запускается в виртуальном окружении.

Процесс загрузки Linux длится, как правило, около минуты. Во многих дистрибутивах в это время отображается индикатор загрузки. В других дистрибутивах, напротив, на экран выводится масса подробной информации о запуске системы. Но такая информация оказывается важна лишь в случаях, когда что-то пойдет не так.

Вход в систему. Как правило, сразу после окончания процесса загрузки отображается графическое окно для входа в систему. Там вы указываете ваше имя пользователя (логин) и пароль. После этого открывается окружение рабочего стола, представляемое в стандартной настольной графической системе вашего дистрибутива — обычно это Gnome или KDE. В одной из следующих глав мы сделаем введение в работу с Gnome.

Но при входе в систему следует представляться не администратором (`root`), а использовать обычный логин! Пользователь-администратор (`root`) имеет в системе неограниченные права. В Linux не принято работать в качестве администратора. С такими правами, как правило, задействуются лишь некоторые программы, применяемые для администрирования системы. Для работы в таком режиме следует указать дополнительно к своему паролю `root`-пароль или (в Ubuntu) подтвердить свой пароль. Об изменении административного пароля и о создании новых пользователей в системе мы поговорим, начиная с раздела 11.4.

Если у вас в системе параллельно установлено несколько окружений Рабочего стола (например, *и* Gnome, *и* KDE), то при входе в систему вы можете выбрать желаемое окружение. В некоторых дистрибутивах при входе в систему также можно настраивать раскладку клавиатуры и язык.

Выход из системы. В меню KDE или Gnome, то есть на панели инструментов, всегда есть команда для выхода из системы. Точное название этой команды отличается в зависимости от дистрибутива и может звучать, например, `SYSTEM USER LOGOUT`. Эта команда завершает все исполняемые программы настольной системы (перед этим необходимо сохранить все открытые файлы!). При выходе вы возвращаетесь к окну для входа в систему — теперь можно либо войти заново, либо перезагрузить компьютер.

Смена пользователя. Как правило, для смены пользователя необходимо выйти из системы, а потом снова туда войти. Но в Gnome и KDE смена пользователя возможна и без выхода. При этом графическая система запускается еще раз, что является довольно ресурсозатратной операцией. Поэтому возможен и быстрый переход между двумя активированными учетными записями.

Вход в систему и выход из нее в текстовом режиме. Многими функциями Linux можно пользоваться и в текстовом режиме. Например, именно в серверных системах обычно принято отказываться от системы X Window или деактивировать ее автоматический запуск. Для работы в таком режиме вход в систему осуществляется из текстовой консоли (см. также главу 5). Для выхода из системы нажмите на клавиатуре Ctrl+D или просто выполните команду `exit`.

Завершение работы Linux. В графических пользовательских интерфейсах предусмотрены специальные команды меню для перезагрузки компьютера, соответствующие параметры также предлагаются в диалоговом окне выхода из системы. В текстовом режиме штатная перезагрузка системы в настоящее время выполняется командой `shutdown -h`. В любом случае для выполнения этой команды необходимы права администратора.

Во многих системах Linux имеется удобная альтернатива для команды `shutdown`. Нужно просто нажать в текстовом режиме сочетание клавиш Ctrl+Alt+Delete. Если вы работаете в X, то предварительно нужно перейти в текстовую консоль, нажав клавиши Ctrl+Alt+F2.

СОВЕТ

Если Linux не реагирует на Ctrl+Alt+Delete, а у вас нет пароля администратора, но компьютер, тем не менее, необходимо перезапустить, выполните перед этим хотя бы команду sync. После этого все буферизованные права на запись файла будут сохранены на жестком диске. Сразу после этого выключите компьютер. Но в любом случае это компромиссное решение, позволяющее всего лишь минимизировать ущерб.

1.2. Клавиатура, мышь и буфер обмена

Важные сочетания клавиш

Набор имеющихся в вашем распоряжении сочетаний клавиш зависит от того, работаете ли вы в графическом режиме или в текстовой консоли. В этом подразделе предполагается, что вы используете именно графический режим. Здесь горячие клавиши определяются для трех уровней программы.

- Система X Window отвечает за реализацию элементарных функций графического окружения. На этом уровне предлагается сравнительно немного сочетаний клавиш (табл. 1.1)

Таблица 1.1. Сочетания клавиш в системе X

Сочетание клавиш	Значение
Ctrl+Alt+Backspace	Принудительно завершает работу графической системы (X Window). В SUSE срабатывает после двойного нажатия. В некоторых дистрибутивах данное сочетание клавиш деактивировано
Ctrl+Alt+Fn	Переход из графической системы в консоль n
Ctrl+Alt+Fn	Переход из текстового режима в консоль n. В большинстве дистрибутивов для возврата в графический режим требуется нажать Alt+F7. В Fedora для этого необходимо использовать Alt+F1

- Системы рабочего стола Gnome и KDE базируются на X. Здесь также определяются некоторые сочетания клавиш. Приятно отметить, что в последние годы здесь удалось достичь определенного единообразия, и, по крайней мере, важнейшие операции осуществляются в Gnome и KDE одними и теми же клавишами (табл. 1.2). Точнее говоря, эти сочетания клавиш берутся из Window Manager (Диспетчер окон), действующего в окружении Рабочего стола.

Таблица 1.2. Важные сочетания клавиш в графических средах рабочего стола (Gnome, KDE)

Сочетание клавиш	Значение
Alt+Tab	Смена активного окна
Alt+F1	Отображение меню Рабочего стола
Alt+F2	Запуск программы
Alt+F3	Отображение контекстного меню актуального окна
Alt+F4	Закрытие окна, соответственно, завершение программы

- Наконец, имеющиеся сочетания клавиш, конечно же, зависят от конкретной программы, которая работает в данный момент. В зависимости от того, что

именно вы делаете — работаете в Интернете в браузере Firefox, пишете письмо в OpenOffice или изменяете программный код в редакторе Vim — всякий раз в вашем распоряжении будут горячие клавиши конкретной программы. Разумеется, в этом подразделе мы их не будем рассматривать.

В программах, которые имеют графический пользовательский интерфейс, для важнейших операций действуют те же сочетания клавиш, что и в Windows. Это касается, например, копирования текста в буфер обмена клавишами Ctrl+C, вставки текста клавишами Ctrl+V либо сохранения документа клавишами Ctrl+S.

Для многих текстовых команд действуют иные условные сочетания, сложившиеся в ходе развития UNIX/Linux. Важные сочетания клавиш такого рода приведены в табл. 5.2.

К сожалению, нельзя утверждать, что приведенные здесь сочетания клавиш действительно будут работать в любом дистрибутиве. Все сочетания клавиш являются настраиваемыми, и некоторые производители не соблюдают тех или иных конвенций. Например, в Fedora и Red Hat для перехода в графический режим применяется не седьмая, а первая консоль. Сочетание клавиш Ctrl+Alt+Backspace по умолчанию деактивируется во все большем количестве дистрибутивов либо срабатывает лишь после двукратного нажатия.

Работа с мышью

В последнее время Linux все активнее ориентируется на условные правила, действующие в Windows и Mac OS. Тем не менее, по-прежнему сохраняются особенности, зависящие от окружения Рабочего стола либо от того, с какой именно программой вы в данный момент работаете. Такие особенности рассмотрены в данном подразделе.

Обычный или двойной щелчок. В Gnome, как и в Windows, для многих операций (например, для открытия файла) применяется двойной щелчок кнопкой мыши. В KDE, напротив, большинство операций по умолчанию выполняются одним щелчком мыши.

Копирование и вставка текста с помощью мыши. Почти во всех программах для Linux с помощью мыши можно копировать фрагменты текста и вставлять их в другой части данного документа либо в другой программе. Для выделения фрагмента текста вы просто проводите по нему мышью, удерживая левую кнопку. Выделенный таким образом текст автоматически копируется в буфер. При нажатии средней кнопки мыши текст будет вставлен туда, где в данный момент находится активный курсор ввода. (В некоторых программах для вставки также может использоваться правая кнопка мыши. Это удобно, если вы применяете мышь всего с двумя кнопками.)

Итак, выделение и копирование текста происходит лишь с помощью мыши, без участия клавиатуры. Если вы привыкнете к такому механизму работы, то у вас просто возникнет вопрос — а почему же в Windows все не так?

Фокус ввода. В некоторых старых X-программах (не имеются в виду программы Gnome и KDE), ввод текста в диалоговом окне возможен лишь в том случае,

когда указатель мыши находится в этом окне. Итак, фокус ввода зависит не от того, какая программа активна в настоящий момент, а от того, где находится указатель мыши.

Такой способ (focus follows mouse, «фокус следует за мышью») в некоторых настольных системах можно задействовать и для работы с окнами. В таком случае уже не обязательно щелкать на окне, чтобы вводить в него информацию. Достаточно просто правильно разместить указатель. Но в таком случае случайный сдвиг мыши может привести к тому, что текст будет вводиться не в то окно или не в ту программу. Поэтому режим focus follows mouse вряд ли широко распространится.

Управление мышью с клавиатуры. Если мышь не работает, то в некоторых дистрибутивах вы можете управлять указателем и с клавиатуры (табл. 1.3). Для этого следует нажать сочетание Ctrl+Shift+Num, которое активирует особый режим работы с клавиатурой. В таком случае вам потребуется клавиатура с цифровым блоком.

Таблица 1.3. Сочетания клавиш для управления мышью с цифрового блока клавиатуры

Клавиши	Значение
4, 6	Перемещение мыши влево или вправо
2, 8	Перемещение мыши вниз или вверх
5	Одиночный щелчок левой кнопкой мыши
+	Двойной щелчок
0	Удержание кнопки мыши (чтобы отпустить кнопку, нажмите 5)
-	Переключение на работу с правой кнопкой мыши (5, + и 0 теперь действуют для правой кнопки)
*	Переключение на левую кнопку мыши

Буфер обмена

Как уже упоминалось выше, любой текст, выделенный мышью, оказывается в оперативно создаваемом буфере обмена. Пока выделение не снято, отмеченный текст можно вставить средней кнопкой мыши в другую программу. Преимущество этого метода заключается в том, что он работает и без клавиатуры. Но есть и недостатки: при каждом новом выделении предыдущее выделение (и содержащий эту информацию буфер обмена) стираются, что очень неудобно. К тому же, далеко не у каждой мыши три кнопки.

Поэтому многие программы (все приложения для Gnome и KDE, Firefox, OpenOffice и др.) содержат дополнительную возможность: как и в Windows, они позволяют копировать текст в буфер обмена с помощью определенного сочетания клавиш, причем такой буфер обмена не зависит от того, какой фрагмент текста выделен в настоящий момент. Как правило, для этой цели применяется то же сочетание клавиш, что и в Windows — Ctrl+C.

Соответственно, вставка содержимого из буфера обмена выполняется сочетанием Ctrl+V или Shift+V.

1.3. Работа с файлами, доступ к внешним носителям данных

Файловая система начинается с корневого каталога /. Если в системе имеется несколько жестких дисков (или сегментов жестких дисков), а также CD- и DVD-приводы, то все данные также встраиваются в дерево каталогов. Например, информация из CD-привода обычно считывается из каталога /media/cdrom. Поэтому в Linux не применяются буквенные обозначения дисков — A:, C: и т. д., как это принято в Windows.

При именовании файлов и каталогов Linux чувствителен к регистру. readme, Readme и README — это три разных файла! Длина имени файла не может превышать 255 символов.

Домашний каталог. После входа в систему вы сразу же оказываетесь в каталоге, который принадлежит именно вам. Этот каталог называется домашним или личным. Все содержащиеся в нем файлы и подкаталоги принадлежат вам. Остальные пользователи (кроме root) не могут ни изменять, ни удалять эти файлы, но, как правило, могут их читать. Домашний каталог сокращенно обозначается тильдой (~). У обычного пользователя Linux домашний каталог располагается по адресу /home/name. У администратора аналогичный адрес таков: /root.

CD, DVD, flash-накопители. В идеале весь дальнейший доступ к внешним носителям данных осуществляется автоматически. Когда диск вставлен в привод, либо flash-накопитель вставлен в USB-вход, на Рабочем столе автоматически появляется соответствующая пиктограмма либо окно файлового диспетчера.

Если автоматическое подключение носителя данных не срабатывает либо если вы работаете с текстовой консолью, то необходимо вручную выполнить команду mount (а по окончании работы — umount, чтобы извлечь носитель из дерева каталогов).

ВНИМАНИЕ

Перед удалением носителя данных, обеспечивающего возможность записи информации, либо перед тем, как вынуть кабель, нужно явно отключить устройство от дерева каталогов. Детали этой операции зависят от конкретной настольной системы, то есть от дистрибутива. Как правило, для этого нужно щелкнуть на соответствующем значке и выполнить команду вида Detach Volume (Отсоединить том) или Unplug or Eject Hardware (Отключить или извлечь устройство). Если этого не сделать, то может возникнуть несовместимость файловой системы носителя данных и потеря информации!

Df. С помощью команды df можно узнать, какие сегменты диска в настоящий момент подключены к файловой системе и сколько свободного места остается в данном сегменте. С помощью параметра -h можно задать единицу измерения дискового пространства равной не Кбайт, а такой, которая лучше соотносится с размерами данного носителя: Мбайт, Гбайт и т. д. В следующем примере к файловой системе подключен не только системный раздел /, но и раздел с данными /myhome. (Кроме того, df учитывает некоторые виртуальные файловые системы, которые не имеют значения для внутрисистемной работы Linux. Не запутайтесь!)

```
user$ df -h
Filesystem          Size  Used  Avail Use%      Mounted on
/dev/sdb6           9,2G  4,0G  4,8G  46%       /
```

```
/dev/sdb5          14G   6,9G  6,2G  53%   /myhome
tmpfs              754M  0     754M  0%    /lib/init/rw
varrun             754M  332K  754M  1%    /var/run
...
```

1.4. Документация по Linux

По Linux имеется практически необъятное количество документации, которая частично поставляется вместе с системой, частично предоставляется в Интернете. Эксперты также могут ознакомиться с исходным кодом, который, как правило, хорошо документирован. В этом разделе сделан краткий обзор важнейших источников информации.

Объем документации настолько велик, что иногда бывает нелегко найти по-настоящему ценную информацию по конкретной проблеме. Очень часто единственный необходимый вам совет может затеряться в массе устаревшей информации, а также в побочных деталях, специфичных для конкретных версий и дистрибутивов. И, конечно же, в запутанных дискуссиях. Кроме того, разумеется, в более выгодном положении оказываются специалисты, владеющие английским языком. Независимо от того, о чем идет речь — об онлайновой помощи по конкретной программе или о техническом описании серверной службы, — переводы, как правило, достать очень нелегко. Зачастую их вообще нет, а то, что удается найти, оказывается неполным или устаревшим материалом.

СОВЕТ

Большая часть документации по Linux содержится в обычных текстовых файлах, HTML- или PDF-документах. Иногда удается найти файлы в формате PostScript. Для чтения таких файлов применяется специальный PostScript-просмотрщик, например Evince или Okular. Как правило, для запуска нужной программы необходимо просто дважды щелкнуть кнопкой мыши на файле в файловом менеджере системы.

Иногда файл бывает заархивирован. Архивные файлы имеют расширение GZ или BZ2. Для извлечения файла из архива нужно выполнить, соответственно, команду gunzip file.gz или bunzip2 file.bz2.

Онлайновая справка. Почти во всех программах, имеющих графический пользовательский интерфейс, при нажатии клавиши F1 выводится онлайновая справка. Если эта функция не работает, убедитесь, что у вас установлены файлы справки. В некоторых крупных программах, таких как Gimp и OpenOffice, подробные файлы справки находятся в собственных пакетах, которые по умолчанию иногда не устанавливаются.

Man и info. При работе со многими текстовыми программами команды `man` *`имя`* или `info` *`имя`* приводят точное описание программы и синтаксическую справку. Тексты `man` и `info` также имеются в справочных системах Gnome и KDE.

Документация по пакетам. В Linux программы устанавливаются в виде пакетов. Пакет зачастую содержит не только все файлы, необходимые для программы, но и документацию. Эта документация, в зависимости от дистрибутива, устанавливается в следующих каталогах:

- Debian, Fedora, Red Hat, Ubuntu — `/usr/share/doc/`*`имяпакета`*;
- SUSE — `/usr/share/doc/packages/`*`имяпакета`*.

Что делать, если вам требуется дополнительная документация по определенной команде, но вы не знаете, к какому пакету относится данная команда? В первую очередь, нужно узнать точные имена файлов интересующей вас команды. Для этого выполняется команда `-a`:

```
user$ which -a cp
/bin/cp
```

Далее вы выясняете, к какому пакету относится конкретный файл. Ход работы зависит от того, какой формат пакетов применятся в вашем дистрибутиве. Следующая команда подсказывает, что `cp` входит в состав пакета `coreutils`:

```
user$ rpm -qf /bin/cp    (Fedora, Red Hat, SUSE)
coreutils-6.4-10
user$ dpkg -S /bin/cp    (Debian, Ubuntu)
coreutils: /bin/cp
```

Форумы и вики-источники по Linux в Интернете. В Интернете имеется множество форумов, вики-источников и других сайтов по Linux, поддерживаемых как фирмами, работающими с Linux, так и энтузиастами. Перечислять их здесь бессмысленно — достаточно набрать в поисковике `fedora forum` или `ubuntu wiki`, и вы обязательно найдете нужные источники. Если вопросы касаются конкретного дистрибутива, то найти ответы на них тем проще, чем популярнее сам дистрибутив. Очевидно, что на форумах идет довольно живой обмен информацией по распространенным дистрибутивам. Отличный пример этого — современные форумы и вики-источники по Ubuntu.

Новостные группы. Новостные группы имеют очень большое значение при изучении Linux. Начинающие пользователи Linux чаще используют форумы и вики-источники, а разработчики часто общаются через почтовые рассылки. Но все же при решении проблем с конфигурацией системы и аппаратным обеспечением вам очень пригодится архив старых новостных сообщений. Лучше всего искать информацию в группах Google: http://groups.google.com.

Проект Linux Documentation. Проект Linux Documentation Project (LDP) ставит своей целью создание максимально полного и централизованного собрания документации по Linux. Действительно, количество информации на сайте http://www.tldp.org/ впечатляет.

Большая часть документов относится к одной из трех категорий: текст-руководство (базовая ориентировочная информация и подсказки), «Часто задаваемые вопросы» (вопросы и ответы) или пособие (электронная книга). Но учитывайте, что многие тексты LDP не поддерживаются и устаревают. Все больше систем и проектов Linux переносят документацию в вики-формат, чтобы в создании и доработке документов могли участвовать все пользователи.

Документация ядра. Множество информации по темам, связанным с оборудованием, содержится в документации по ядру Linux. Она является частью кода ядра. Документация по наиболее актуальной версии ядра расположена в Интернете по следующему адресу: http://www.kernel.org/doc/Documentation/.

RFC. Аббревиатура RFC означает Request for Comment (Запрос комментариев). Такой аббревиатурой обозначаются документы, подробно описывающие различные протоколы (в частности, TCP, IP, FTP, PPP и т. д.). Возникновение несколько странной аббревиатуры RFC понятно из истории создания таких документов: как правило, они составлялись не специалистом, организацией или фирмой, а получались в результате дискуссий, зачастую довольно длительных. В таких источниках предлагается сложная техническая информация. Хорошей подборкой таких документов является сайт http://www.faqs.org/rfcs/.

2 Gnome

Когда вы работаете в Windows или Mac OS X, вам предлагается пользовательский интерфейс, который является частью операционной системы. В Linux ситуация иная: операционная система отвечает лишь за выполнение базовых функций. За работу с пользовательским интерфейсом отвечают более высокоуровневые программы. По разным причинам в Linux со временем сформировалось несколько графических систем для работы на настольном компьютере. Важнейшие из них — Gnome, KDE, Xfce и LXDE.

Независимо от того, с какой именно настольной системой вы работаете, набор ее основных функций меняется мало. К ним относятся:

- управление Рабочим столом, на котором, как правило, располагается одна или несколько панелей. В этих панелях находятся меню Пуск, панель задач и другие мини-программы;
- диспетчер окон, отвечающий за управление окнами (он позволяет сменить активное окно, переместить окно и т. д.);
- различные прикладные и конфигурационные программы.

Основной темой данной главы являются базовые функции Gnome 3.*n*. Вместе с Gnome также устанавливаются многочисленные программы, которые функционально и внешне приспосабливаются к конкретной настольной системе. Так или иначе, в этой главе мы сможем рассмотреть лишь малую толику таких приложений. Некоторые другие программы изучаются в тематических главах. При написании этой главы я использовал дистрибутив Fedora 17 с Gnome 3.4. Пока мне еще не представилось шанса протестировать новую версию Gnome 3.6, в частности, для описания диспетчера файлов, который в этой версии значительно изменился.

Gnome 3. В Gnome 3 разработчики радикально изменили Рабочий стол системы (рис. 2.1). Основным новшеством является совершенно новая оболочка Gnome (Gnome Shell). Это компонент программы, отвечающий за запуск программ и управление окнами.

На вид система получилась очень привлекательной. По-моему, Gnome предлагает самый красивый на настоящее время Рабочий стол для Linux (разумеется, это совершенно субъективная оценка).

Но с функциональной точки зрения все не так однозначно. Из-за отказа от привычных принципов работы с интерфейсом, а также из-за отсутствия возможности конфигурировать систему, проект Gnome разочаровал многих Linux-профессионалов.

Изобретатель Linux Линус Торвальдс и разработчик файловых систем Теодор Тсо открыто выразили свое негодование по этому поводу:

- https://plus.google.com/106327083461132854143/posts/SbnL3KaVRtM;
- https://plus.google.com/117091380454742934025/posts/VjbFCa7X5NJ.

Рис. 2.1. Рабочий стол Gnome

Разработчики Gnome оправдываются тем, что хотели облегчить для новичков обращение с Рабочим столом. Правда, сомнительно, что им это удалось. Компания Canonical убеждена в противоположном. Она заменила некоторые важные элементы Gnome собственной разработкой под названием Unity. Подобным образом поступили разработчики Linux Mint. Теперь Mint предлагается в двух вариантах, один из которых основан на Gnome 2.*n* (точнее, на MATE), а другой вносит в Gnome 3.*n* настолько значительные изменения, что эта версия Gnome становится практически неузнаваемой.

Еще одним основанием для критики является зависимость новой системы от драйверов для поддержки 3D-графики. Если подходящий драйвер отсутствует (такая ситуация возникает, например, на компьютерах с современными графическими картами NVIDIA, которые не работают без установки проприетарного драйвера, а также на виртуальных машинах), то Gnome запускается в ограниченном режиме. Работа в этом режиме напоминает Gnome 2.*n* (тем не менее, третья версия

со второй несовместима!) и значительно отличается от Gnome 3.*n*. Правда, решение для этой проблемы уже найдено: в Fedora 17 и openSUSE 12.2 есть дополнительные графические библиотеки, которые позволяют использовать Gnome 3.*n* и без драйверов для 3D-графики. Видимо, их примеру вскоре последуют и другие дистрибутивы.

СОВЕТ

Не стоит сразу отказываться от Gnome 3 из-за этих критических отзывов — дайте ему шанс! Многие концепции Gnome 3 хорошо продуманы и обеспечивают эффективную работу, к ним только надо привыкнуть. Кроме того, Gnome 3 очень легко модифицировать с помощью подключаемых расширений: один щелчок кнопкой мыши в браузере — и такое расширение установится самостоятельно, обогатив Gnome новыми функциями.

2.1. Организация Рабочего стола

Вход в систему и выход из нее. Прежде чем приступать к работе в Gnome, нужно войти в систему под вашим именем пользователя (логином) и паролем. Если на вашем компьютере установлено несколько систем Рабочего стола, то в окне для входа в систему вы также можете выбрать желаемую среду.

Чтобы вновь выйти из системы, щелкните кнопкой мыши в правом верхнем углу панели Gnome, где указан ваш логин, и выполните команду Log out (Выйти из системы). А что делать, если вы хотите не просто выйти из системы, а выключить и перезапустить компьютер? Такой пункт меню разработчики Gnome спрятали поглубже. Команда Turn off (Выключить) появляется после нажатия клавиши Alt!

Эта глупая игра в прятки устраняется так же, как и многие другие сомнительные умолчания в Gnome 3 — с помощью инструмента Gnome Tweak Tool (см. раздел 2.4). В Gnome 3.6 есть еще одно радикальное новшество, связанное с такими проблемами: пока в системе работает лишь один пользователь, в меню присутствует только запись Turn Off (Выключить) и нет записи Log out (Выйти из системы).

Смена пользователя. Когда с одним компьютером работает несколько пользователей, совсем не обязательно возникает такая ситуация: один пользователь выходит из системы для того, чтобы второй мог быстренько проверить почту. Скорее, для этого в меню User Name (Имя пользователя) нужно будет выполнить «оперативную смену пользователя». Внутри системы для каждого пользователя запускается собственный X-сервер. Поэтому для параллельной работы одновременно под несколькими учетными записями требуется масса ресурсов.

Панель. Все управление Рабочим столом сосредоточено на панели, которая прочно обосновалась вдоль верхнего края экрана. Здесь находится кнопка Activities (Действия), ярлык той программы, которая активна в настоящий момент, показывается текущее время, а также (в правой части панели) — различные статусные пиктограммы и меню. Правее располагается уже упоминавшееся меню User Name (Имя пользователя), в котором вы можете выйти из системы, изменить системные настройки или сменить активного пользователя. Сама рабочая область — не считая открытых окон, которые могут здесь находиться, — совершенно пуста. Отображение ярлыков на Рабочем столе не предусмотрено.

Меню приложений. Отдельные программы Gnome дополнительно предлагают свои важнейшие команды и в так называемом меню приложений (рис. 2.2). Чтобы перейти в это меню, нужно нажать на панели имя той программы, которая активна в настоящий момент. В некоторых программах меню приложения даже заменяет обычное меню. Означает ли это, что и в Gnome появится центральное меню, чье возникновение в Ubuntu вызвало столько дискуссий?

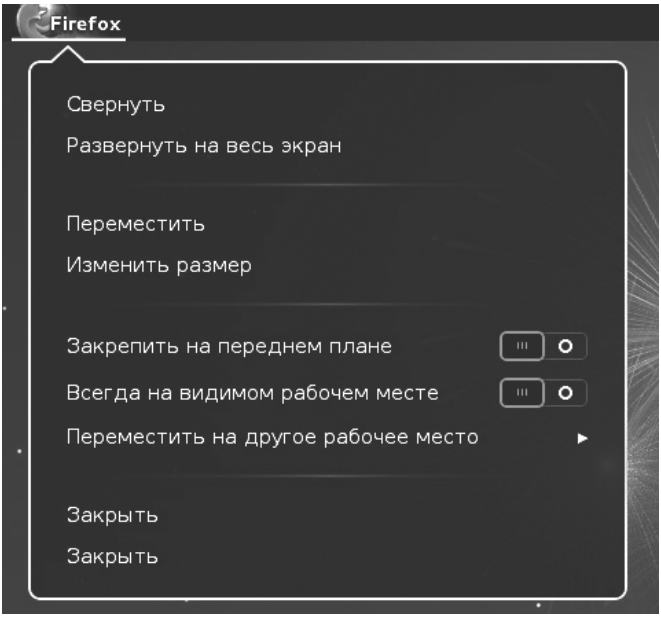


Рис. 2.2. Меню приложений в браузере Gnome

Действия. Если щелкнуть на кнопке Действия, переместить указатель мыши в верхний левый угол экрана, нажать клавишу Windows или Alt+F1, откроется так называемая перспектива с действиями. По умолчанию в этой перспективе слева отображается панель с ярлыками часто используемых программ, а также всех программ, которые работают в данный момент. Справа предлагается предварительный просмотр активных рабочих окон. При этом в системе также имеется перспектива Art Exposé, где отображаются все окна рабочей области. Теперь вы можете, например, переносить окна в новую рабочую область, перестраивать ярлыки часто используемых программ в панели ярлыков у левого края экрана и т. д.

Поисковая функция. В правом верхнем углу перспективы действий автоматически активируется поисковое поле. Если ввести туда с клавиатуры поисковый запрос, то Gnome заменит общий вид всех окон на результаты поиска. При поиске учитываются программы, модули системных настроек, каталоги, контакты и последние использовавшиеся файлы. Желаемый объект можно выбрать либо с помощью мыши, либо клавишами управления курсором ↑ и ↓. (К сожалению, клавиши управления курсором ← и → при этом не используются — они применяются для перемещения курсора при вводе поискового запроса.)

Поисковая функция — несомненно, самое замечательное новшество в Gnome 3.*n*. Если вы, например, хотите быстро открыть Gimp, нажмите просто клавиши Windows GI Enter. Если вы к этому привыкли и наизусть помните, с каких букв начинаются названия основных программ, то запуск приложений происходит исключительно быстро и эффективно.

Учитывайте, что клавиши Windows xxx Enter активизируют уже работающую программу, а не запускают новый экземпляр. Это целесообразно в большинстве случаев, но не всегда: если вы, например, хотите не активировать уже работающее окно терминала, а открыть новое такое окно, то нужно нажать Ctrl+Enter либо щелкнуть кнопкой мыши на ярлыке терминала, удерживая Ctrl.

Запуск программ. Чтобы запустить программу, названия которой вы не знаете, щелкните в перспективе Действия на кнопке Приложения. После этого откроется алфавитный список ярлыков всех установленных программ. Как правило, этот список очень длинный и неудобный для изучения. Чтобы исправить ситуацию, можно либо задать поисковый запрос, либо отфильтровать результаты поиска по типу программы (например, INTERNET).

Смена активной программы. Работа с Gnome на клавиатуре не кажется интуитивно понятной. На первых порах особенно раздражает тот факт, что сочетание клавиш Alt+Tab не позволяет переключаться между окнами, а осуществляет переход между программами. (Такая концепция уже довольно давно используется в Mac OS X, но даже Apple не удастся меня убедить, что это действительно хорошая идея.) Когда нужная программа найдена, вы можете выбрать интересующее вас окно сочетанием клавиш Alt+клавиши управления курсором. Но на это требуется терпение! Гораздо проще переключаться между окнами сочетанием клавиш Alt+Esc. Это сочетание работает, в сущности, так же, как ранее работала Alt+Tab. Не менее удобно сочетание Alt+^, которое позволяет переключаться между окнами той программы, которая активна в настоящий момент. Итак, для перехода между окнами у нас есть *три* сочетания клавиш, хотя раньше все прекрасно работало всего с одним сочетанием клавиш.

Панель инструментов. В Gnome нет специальной панели задач или панели приложений, которая постоянно отображалась бы в окне. Эту роль играет вертикальная панель с ярлыками, расположенная по левому краю перспективы Действия. В англоязычных источниках эта панель называется Dock или Dash.

В верхней части панели инструментов по умолчанию находятся некоторые программы, которые, на взгляд разработчиков Gnome, используются особенно часто. В нижней части панели инструментов располагаются ярлыки всех программ, запущенных в данный момент, — если эти ярлыки не находились в панели инструментов по умолчанию.

Чтобы удалить программу с панели инструментов, выберите в контекстном меню команду Удалить из избранного. Чтобы добавить программу на панель инструментов, перетащите ее мышью из перспективы Приложения на панель инструментов. Кроме того, можно выполнить в контекстном меню уже работающей программы команду Добавить в избранное.

Статусная область. В нижней части экрана располагаются статусные сообщения и уведомления. Как правило, статусная панель на экране не отображается. Чтобы

просмотреть содержащуюся в ней информацию, нужно подвести указатель мыши к правому нижнему углу экрана.

Окна. В Gnome 3.*n* отсутствуют оконные кнопки Свернуть и Развернуть. Чтобы развернуть окно, переместите его вплотную к верхнему краю экрана или дважды щелкните на строке с названием окна.

Как и в Windows, окна можно размещать в левой или в правой половине монитора, захватывая окно мышью за левую или правую сторону.

Рабочие области. Рабочие области позволяют распределять окна работающих программ по нескольким виртуальным Рабочим столам, а потом переходить между этими Рабочими столами. Так облегчается работа и улучшается обзор, даже если одновременно открыто несколько окон. Например, можно запустить в собственной рабочей области программу для обработки изображений Gimp. Тогда в одной рабочей области будет находиться сколько угодно открытых окон Gimp, а все остальные окна будут в другой рабочей области.

Управление рабочими областями в Gnome 3.*n* происходит динамически. По умолчанию есть только одна рабочая область. В перспективе действий можно переместить окно во вторую рабочую область. Если уже есть две рабочие области, Gnome предлагает третью — сначала совершенно пустую. Независимо от того, сколько рабочих областей вы используете, в вашем распоряжении обязательно будет еще одна.

Что касается постоянно востребованных окон, то существует возможность пометить их так, что они будут отображаться не в одной рабочей области, а во всех. Для этого откройте меню окна правой кнопкой мыши или сочетанием клавиш Alt+Пробел и установите флажок Всегда в видимой рабочей области.

Апплеты. Апплеты, отображаемые в правой части панели, практически не приспособлены для внесения изменений. Нет никакого диалогового окна, в котором можно было бы добавлять новые апплеты. Если все же потребуется добавить их, откройте страницу https://extensions.gnome.org/, где можно скачать и активировать дополнения для Gnome. Апплеты Gnome 2.*n* несовместимы с апплетами Gnome 3.*n* и больше использоваться не могут.

Сочетания клавиш. В табл. 2.1 приведены основные сочетания клавиш, применяемые в Gnome 3.*n*. Дополнительные сочетания клавиш находятся в системных настройках, модуль Клавиатура, диалоговый список Горячие клавиши.

Таблица 2.1. Важные горячие клавиши в Gnome 3.0

Комбинация клавиш	Значение
(Windows или Alt)+F1	Переход между стандартной перспективой и обзором Рабочего стола (Expose). Кроме того, вы попадаете в перспективу Expose, переместив указатель мыши в верхний левый угол окна. Теперь с клавиатуры можно вводить поисковые запросы
Alt+F2	Запускается программа, имя которой вы укажете
Alt+Tab	Переход между программами (а не окнами)
Alt+Esc	Переход между всеми окнами (ранее это делалось с помощью Alt+Tab)
Alt+^	Переход между окнами программы, активной в настоящий момент

Продолжение ⇩

Таблица 2.1 (продолжение)

Ctrl+Alt+Tab	В стандартной перспективе перемещает фокус ввода в панели и позволяет работать с элементами панели. В перспективе Рабочего стола позволяет переходить между различными настольными элементами, то есть, панелью, боковой панелью (Dash), окнами, рабочими областями и др.
Ctrl+Alt+Tab+↓+↑	Переход между рабочими областями
Shift+Ctrl+Alt+Tab+↓+↑	Перенос данного окна в следующую рабочую область

2.2. Файловый менеджер

Программа Files (ранее — Nautilus) — это файловый менеджер для Рабочего стола Gnome. Программа не только предоставляет доступ к файлам и каталогам, но и обеспечивает доступ к внешним носителям данных и к сетевым каталогам. В этом разделе рассказано о работе с файловым менеджером, но не будем углубляться в подробности того, как в Linux осуществляется управление файлами. О том, что такое ссылки, как помечаются скрытые файлы и как в Linux функционируют права доступа, вы узнаете в главе 7.

Gnome 3.6. Внешний вид и принципы работы файлового менеджера Gnome в версии 3.6 значительно изменились. Весь этот подраздел написан на основе тестов, которые я провел с бета-версией Gnome 3.6. Итак, как я уже упоминал выше, программа Nautilus теперь называется просто Files. Можете себе представить, каково пытаться найти в Google по такому поисковому запросу что-либо интересующее вас! Но новое название — еще полбеды. В типичной гномовской манере разработчики радикально упростили программу. В итоге: программа хорошо прибрана и удобна в использовании, но потеряла массу полезных функций.

- В программе больше нет обычного меню — только меню с инструментами и меню приложений. Закладки теперь можно вызывать только из боковой панели.
- Больше нет специальной перспективы для компактного просмотра файла.
- Упразднен параллельный просмотр двух каталогов, открывавшийся клавишей F3.
- Удален просмотр дерева каталогов в боковой панели.
- По умолчанию каталоги теперь не сортируются раньше, чем файлы (к счастью, эту несуразицу можно устранить с помощью программных настроек).

Среди читателей сайта http://www.omgubuntu.co.uk/ уже около 18% придерживаются мнения, что эти изменения к лучшему. Разработчики дистрибутива Linux Mint сделали соответствующие выводы и представили ответвление диспетчера файлов, которое назвали Nemo. В будущих версиях Linux Mint Nemo должен прийти на смену Nautilus.

Отображение файлов. Чтобы запустить файловый менеджер, проще всего дважды щелкнуть на его ярлыке на панели инструментов (в перспективе Действия). По умолчанию файловый менеджер отображает содержимое выбранного каталога в перспективе Значки. Каждый файл представлен пиктограммой. В случае с кар-

тинками и файлами некоторых других типов эта пиктограмма одновременно является эскизом для предварительного просмотра содержимого. По умолчанию предварительный просмотр возможен только при работе с локальными файлами (но не файлами из сетевых каталогов), которые по размеру не превышают 10 Мбайт. Эти параметры можно изменить в меню инструментов с помощью команд Правка ▸ Настройки ▸ Предварительный просмотр.

Чтобы экземпляр для предварительного просмотра не приходилось каждый раз создавать заново, файловый менеджер сохраняет такие изображения в каталоге `~/.thumbnails`. Этот каталог используют и многие другие программы Gnome.

С помощью сочетаний клавиш Ctrl+1 или Ctrl+2 можно переходить из перспективы Значки в перспективу Таблица.

Боковая панель. Вдоль левого края окна обычно располагается боковая панель, позволяющая переходить в важнейшие каталоги (перспектива Места). Но в боковой панели можно отображать и дерево каталогов. Клавиша F9 вновь возвращает на экран боковую панель.

Смена каталога. В панели символов находится несколько кнопок, с помощью которых можно быстро перейти в каталоги, расположенные выше по иерархии. Кроме того, эти кнопки позволяют отобразить имя наиболее актуального каталога (directory path). В качестве альтернативы файловый менеджер может показать здесь и полный путь к каталогу (Ctrl+L). Так упрощается и быстрый ввод другого каталога.

Вкладки. С помощью команды Файл ▸ Новая вкладка или сочетанием клавиш Ctrl+T открывается новый лист диалогового окна (dialog sheet). Такие листы особенно удобны в случаях, когда вы хотите копировать файлы из одного каталога в другой либо вырезать их в одном и вставить в другом каталоге. Кроме того, можно менять активный лист диалогового окна в ходе операций перетаскивания.

Открытие файлов. В большинстве случаев, чтобы открыть файл, на нем достаточно дважды щелкнуть кнопкой мыши. Файловый менеджер автоматически запускает соответствующую программу. Если данный тип файла файловому менеджеру не известен, то щелкните на файле правой кнопкой мыши и выберите команду Открыть с помощью. Откроется диалоговое окно, где будут перечислены уже установленные программы, возможно, подходящие для этой цели.

Для обработки файлов некоторых типов может подходить сразу несколько приложений. Например, файлы изображений можно открывать в просмотрщике изображений, Gimp или Firefox. Одна из этих программ считается заданной по умолчанию и открывается двойным щелчком кнопкой мыши. Если вы хотите изменить это умолчание, то щелкните на файле правой кнопкой мыши, выполните Свойства ▸ Открыть с помощью и выберите желаемую программу. После этого настройка будет действовать для всех файлов с одинаковым расширением, то есть, например, для всех PNG-файлов.

Перемещение и копирование файлов. Чтобы копировать уже выделенные файлы, нажмите Ctrl+C, а чтобы вырезать — Ctrl+X. Затем, чтобы вставить интересующий вас файл на новое место, нажмите Ctrl+V (вырезанные файлы сначала удаляются по исходному адресу).

Все будет гораздо проще, если вместо клавиатуры воспользоваться мышью и перенести файлы из одного окна файлового менеджера в другое методом перетаскивания. При этом файлы, как правило, именно переносятся, а не копируются. Исключение составляют операции перетаскивания, осуществляемые между различными носителями данных. Например, речь может идти о переносе файла с CD или из сетевого каталога в локальную файловую систему. В таком случае рядом с указателем мыши появляется маленький «плюсик», подчеркивающий суть операции.

Когда вы хотите именно скопировать, а не переместить файл, нажмите во время перетаскивания клавишу Ctrl. Если вы сами хотите задать режим переноса, нажмите клавишу Alt. Отпустив кнопку мыши, вы сможете копировать файл, переместить его или создать связь (ссылку).

Поиск файлов. Кнопка Поиск позволяет указать в адресной строке поисковый запрос. После этого файловый менеджер выведет список всех файлов, содержащих такой текст в имени файла. После окончания поиска полученные результаты можно отсортировать по типу документа или по каталогу. Кроме того, для поиска можно применить программу Искать файлы. Имя этой программы — `gnome-search-tool`. Программа учитывает гораздо больше второстепенных поисковых критериев, но работает довольно медленно.

Гораздо быстрее можно отыскать файлы с помощью настольного поисковика. Вообще, лишь в некоторых дистрибутивах пакеты для такого поиска устанавливаются по умолчанию. В дистрибутиве Fedora соответствующие пакеты называются `tracker` и `tracker-ui-tools`. Кроме того, есть еще пакет `tracker-preferences`, позволяющий настраивать различные поисковые параметры. Чтобы запустился поисковик, нужно заново войти в систему. Он сохраняет индексные файлы в каталоге `~/.cache/tracker`. Сам локальный поиск запускается клавишей Windows (программа `tracker-needle`). Если получено несколько результатов поиска, то поиск можно ограничить типом файла.

Скрытые файлы. Все файлы и каталоги, чье имя начинается с точки, в Linux считаются скрытыми. Это означает, что по умолчанию они не отображаются в файловом менеджере и диалоговых окнах для выбора файлов. Скрытые файлы часто содержат конфигурационные настройки или другие данные, которые не следует изменять напрямую. Напрямую изменять скрытые файлы и каталоги бывает целесообразно лишь в исключительных случаях (например, когда вы хотите сделать резервную копию папок с электронной почтой в каталоге `~/.thunderbird`). Чтобы такие файлы и каталоги стали отображаться в файловом менеджере, выполните команду Вид ▸ Показать скрытые файлы или нажмите Ctrl+H.

Права доступа. Чтобы конкретный пользователь не мог читать или изменять сразу все файлы, Linux сохраняет для каждого файла и каталога имя его владельца и права доступа. Концепция, лежащая в основе такого механизма, подробно обсуждается в разделе 7.6. Чтобы изменить данные о владельце файла или о правах доступа, щелкните на файле правой кнопкой мыши и выполните команду Свойства ▸ Права доступа.

Удаление файлов. Когда вы удаляете файлы и каталоги, они сначала оказываются в корзине. Чтобы отобразить содержимое корзины, нужно дважды щелкнуть

на соответствующем ярлыке на Рабочем столе либо в боковой панели файлового менеджера. Только после того, как вы выделите здесь интересующие вас объекты и нажмете Delete, файлы будут удалены окончательно. Чтобы сразу удалять файлы без помещения их в корзину, нажимайте Shift+Delete.

Внешние носители данных. Если вы вставляете в привод CD или DVD либо подключаете носитель данных через USB, Firewire или eSATA, то автоматически открывается новое окно файлового менеджера, в котором отображаются файлы, содержащиеся на этом носителе. Соответствующие параметры находятся в области системных настроек в модуле Подробности ▸ Съемный диск.

Не забывайте, что внешние жесткие диски и флешки нужно обязательно виртуально «извлекать» из компьютера, и лишь потом отключать кабель. Для этого нажмите специальную кнопку Извлечение на боковой панели.

Сетевые функции

Совместное использование Windows. Команда Обзор сети в боковой панели открывает перспективу, в которой через несколько секунд отображаются ярлыки всех обнаруженных сетей. На практике здесь обычно удается обнаружить только сеть Windows (Windows Network). После двойного щелчка вы переходите в следующую перспективу, где показаны все обнаруженные сети Windows.

После следующего двойного щелчка отображаются все компьютеры, видимые в данной сети.

Еще один двойной щелчок — и вы увидите, сколько ресурсов имеется в распоряжении у конкретного компьютера.

Если сетевой каталог защищен паролем, то для работы с ним вы должны указать логин и пароль. Тогда вы сможете долговременно сохранить эти учетные данные в базе паролей Gnome. Чтобы не проделывать каждый раз заново весь путь до сетевого каталога — а такой путь порой бывает довольно сложным, — сделайте на него закладку с помощью сочетания клавиш Ctrl+D.

Если файловый менеджер может использовать сетевой каталог и без пароля, он всегда выбирает именно этот вариант. Правда, в определенных случаях такой вариант не идеален. В зависимости от того, как сконфигурирован сервер Windows или Samba, файловый менеджер может отобразить на данном этапе просто пустой каталог. А войти в систему под своим именем через графический интерфейс вы уже не сможете.

Что нужно сделать: нажмите Ctrl+L и введите свой логин в открывшийся путь. Правильная запись такова: `smb://login@servername/directoryname`.

Если файловый менеджер не найдет сервера Windows, то причиной такой ошибки, скорее всего, окажется чересчур непроницаемый сетевой экран между вашим компьютером и компьютером Windows. Часто не функционирует лишь разрешение имен. В таком случае нажмите Ctrl+L и укажите адрес `smb://servername`. Аналогичным образом можно устанавливать связь с другими серверными службами (AFP, FTP, WebDAV, SSH и т. д.).

В табл. 2.2 приведены важнейшие адреса и протоколы. Здесь также упомянуты специальные адреса `computer:` и `trash:`.

Таблица 2.2. Специальные адреса

Адрес	Результат
computer:	Список всех носителей данных
afp://user@hostname	Доступ к AFP-серверу (Apple)
ftp://hostname	Доступ к FTP-серверу
network:	Применение в качестве общего обозревателя сети
sftp://hostname	Доступ к SFTP-серверу (протокол SSH)
smb:	Применение в качестве обозревателя сети Windows
smb://hostname	Доступ к сетевым каталогам компьютера Windows
trash:	Корзина (удаленные файлы)

Еще один способ открытия сетевых каталогов — это команда Подключение к серверу на панели инструментов. Обратите внимание на то, что в некоторых версиях Gnome требуется указывать доменное имя сети Windows в верхнем регистре. В Gnome 3.6 эта ошибка, к счастью, устранена, регистр больше значения не имеет.

Если у вас установлено расширение `nautilus-share`, то файловый менеджер поможет вам предоставить для совместного использования в сети и ваши собственные каталоги. Но это возможно лишь при условии, что локальный сетевой экран допускает использование данного компьютера в качестве Samba-сервера и если вы предоставляете ваш каталог без пароля. В некоторых дистрибутивах к тому же следует предварительно настроить базовую конфигурацию сервера Samba. Такая ситуация, например, возникает в SUSE, где для этого применяется YaST-модуль Сетевые службы ▸ Сервер Samba.

GVFS. Для доступа к сетевым каталогам в Gnome применяется GVFS (Gnome Virtual File System, виртуальная файловая система Gnome). Внешние каталоги подключаются к дереву каталогов как поддиректории от `~/.gvfs`. В диспетчере файлов, а также в диалоговых окнах Gnome для выбора файлов внешние сетевые каталоги отображаются в боковой панели (если вы их не видите — нажмите F9).

Плагины

Функции файлового менеджера можно расширять с помощью плагинов. В большинстве дистрибутивов предоставляются пакеты для некоторых плагинов, но по умолчанию эти пакеты не устанавливаются. Поищите в программе управления пакетами вашего дистрибутива имя `nautilus`, установите желаемые пакеты, а потом снова войдите в Gnome. Здесь я расскажу о некоторых пакетах, приведенные имена действительны для Debian и Ubuntu.

- `nautilus-image-converter` и `nautilus-image-manipulator`: оба плагина позволяют поворачивать изображения или изменять их размер (рис. 2.3).
- `nautilus-compare`: с помощью этого плагина можно сравнить два или более предварительно отмеченных текстовых файла. Разница между файлами будет графически представлена в программе Meld.
- `nautilus-open-terminal`: этот плагин позволяет, щелкнув правой кнопкой мыши, открыть на Рабочем столе окно терминала.

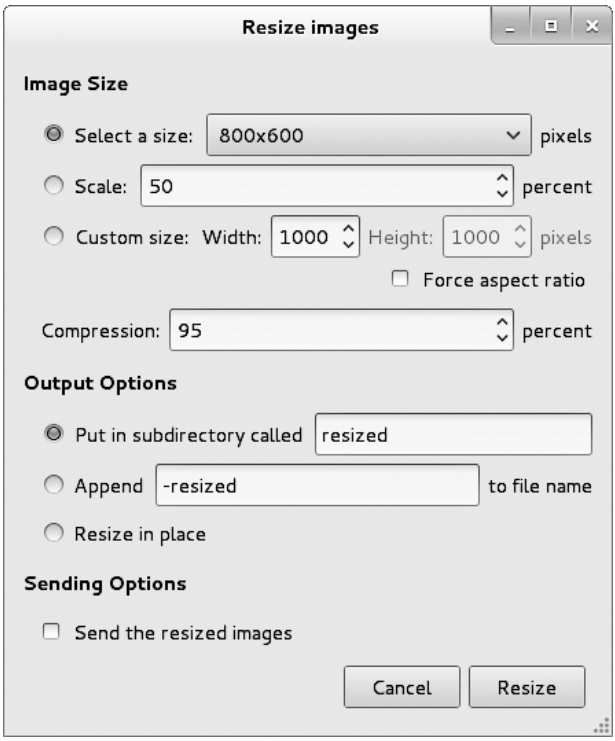


Рис. 2.3. Изменение размера изображения с помощью nautilus-image-manipulator

- `nautilus-pastebin`: с помощью данного плагина можно загружать текстовые файлы на сервис Pastebin.
- `seahorse-nautilus`: плагин позволяет шифровать выбранные файлы с помощью контекстного меню.
- `nautilus-dropbox`: этот плагин помогает синхронизировать каталог `Dropbox` с вашим аккаунтом на сервисе Dropbox.

Запись CD и DVD

CD и DVD с данными. Нет ничего проще, чем записать отдельные файлы или целые каталоги на CD или DVD. Окно Создание CD/DVD открывается автоматически, как только вы вставляете в привод пустой CD или DVD. Если этого не произойдет, запустите программу Brasero и щелкните на кнопке Data Project (Проект с данными). Теперь скопируйте из окна файлового менеджера файлы или каталоги, которые требуется сохранить на диск, в окно записывающей программы (это делается методом перетаскивания). В диалоговом окне Burn (Запись) установите скорость записи — и вперед!

После записи Brasero автоматически проверяет, нет ли ошибок на новом диске. Если вы не хотите выполнять этот шаг, то деактивируйте в пользовательском

интерфейсе Brasero плагины контрольных сумм (выполните команду Edit ▸ Plugins (Правка ▸ Плагины)).

Копирование CD/DVD. Чтобы копировать CD или DVD, вставьте его в привод, вновь запустите программу Brasero и выполните команду Copy CD/DVD (Скопировать CD/DVD диск). Brasero запишет содержимое CD/DVD в ISO- или TOC-файл. Когда считывание данных завершится, вы сможете записать их на пустой CD или DVD.

ISO- и TOC-файлы. В ISO- или TOC-файле содержится полное содержимое CD или DVD. Чтобы записать такой файл, щелкните на нем правой кнопкой мыши и выполните команду Write on CD/DVD (Запись на CD/DVD).

Кстати, предварительно можно просмотреть содержимое ISO-файла. Двойным щелчком на файле вы открываете менеджер архивов, который работает с ISO-файлом примерно как с ZIP-архивом. Правда, изменять содержимое ISO-файла вы не можете.

ISO-файлы можно создавать самостоятельно. Данный процесс напоминает запись CD или DVD, но в качестве итогового носителя выберите файл образа (image file).

AudioCD. Если вы хотите записать аудиодиск, содержащий MP3- или OGG-файлы, которые можно слушать на обычном CD-проигрывателе, снова запустите Brasero и создайте новый аудиопроект (Audio Project). Поместите в этот проект желаемые аудиофайлы. Это можно сделать методом перетаскивания, кнопкой Add (Добавить) или с помощью файлового обозревателя. Он интегрирован в боковую панель и вызывается клавишей F7. Чтобы начать запись, нажмите кнопку Burn (Запись). Этот процесс длится немного дольше, чем обычная запись, поскольку аудиофайлы предварительно приходится преобразовать в WAV-формат.

2.3. Стандартные программы Gnome

В этом разделе я расскажу о некоторых программах, которые обычно поставляются вместе с настольной версией Gnome. Поскольку проект Gnome, наконец, окончательно перешел к обозначению программы по ее функции (например, программа-браузер называется Web), я буду указывать в скобках прежнее название или внутрисистемное имя программы.

Boxes. Начиная с Gnome 3.6, программа Boxes должна стать неотъемлемой частью этой среды. Boxes обеспечивает настройку и эксплуатацию виртуальных машин с помощью KVM. Программа Boxes ориентирована на пользователей Linux, желающих быстро и без проблем запустить на виртуальной машине систему Windows или другой дистрибутив Linux. Boxes уже присутствует в стандартной сборке Fedora 17, но эта версия Boxes еще довольно нестабильна и плохо приспособлена для практического применения.

Файловый архив (file-roller). Для передачи файлов по электронной почте или для создания резервных копий часто бывает целесообразно архивировать несколько файлов или целый каталог. При этом вам пригодится так называемый менеджер архивов. Программа запускается, как правило, двойным щелчком на файле архива.

Менеджер архивов отобразит этот файл так, как будто перед нами обычный каталог.

Если вы хотите сделать быстрый обзор по всем файлам, то выполните команду Вид ▸ Отобразить все файлы. Чтобы распаковать весь архив, нажмите кнопку Распаковать. Чтобы создать новый архив, нажмите сочетание клавиш Alt+F2 и выполните команду `file-roller`. Теперь путем обычного перетаскивания сюда можно переносить файлы и целые каталоги.

Документы (gnome-documents). Программа Документы дает обзор последних использовавшихся офисных и PDF-файлов. Если настроить в модуле Онлайновые учетные записи (Online Accounts) аккаунт Google, то можно синхронизировать и файлы из Google Docs. Но когда я тестировал эту программу, она показалась мне несколько сырой.

Удаленное обслуживание (vino-preferences). Если у вас возникает проблема с компьютером (например, если какая-то программа работает неправильно), вы можете обратиться за помощью по электронной почте или по телефону. Но, по моему опыту, такие запросы о помощи довольно неудобны. «Просто щелкните по кнопке xy!» — «А где эта кнопка?» — «Вы также можете воспользоваться командой меню «ABC ▸ EFG» — «В какой программе?»

Есть очень удобный вариант решения подобных проблем. Благодаря возможности удаленного управления консультант на время получает полный контроль над вашим компьютером. На экране своей машины он видит все содержимое вашего Рабочего стола и может работать со всеми вашими программами с помощью мыши и клавиатуры. Чтобы запустить в Gnome удаленное управление для получения консультации, выполните программу совместного использования рабочей области (эта программа называется `vino-preferences`). Ваш консультант может воспользоваться любым VNC-клиентом. Например, подойдет программа для Gnome `vinagre`.

К сожалению, такое удаленное управление функционирует лишь тогда, когда оба компьютера работают в одной локальной сети или имеют однозначные IP-адреса. Если же компьютеры соединены ADSL- или LAN-роутером, то установить удаленное управление в Gnome не удается. В таких случаях вам могут помочь коммерческие программы. Мне, например, понравилось работать с TeamViewer. Эта программа имеет версию для Linux, которая бесплатна для личного использования (http://www.teamviewer.com/en/index.aspx).

Работа с жесткими дисками (baobab). Если вы хотите узнать, в каких из ваших каталогов находятся самые объемные данные, вам поможет программа для анализа дискового пространства (Disk Usage Analyzer). В Gnome она называется baobab (Анализатор использования дисков). Программа выводит наглядную схему того, сколько информации содержится в тех или иных каталогах или подкаталогах (рис. 2.4). Для создания такого графика должны быть считаны все подкаталоги.

Чтобы создать описанный график, необходимо получить информацию обо всех подкаталогах. Для этого щелкните на кнопке Сканировать домашнюю папку или Сканировать файловую систему, если хотите проанализировать всю файловую систему (на это может потребоваться довольно много времени!).

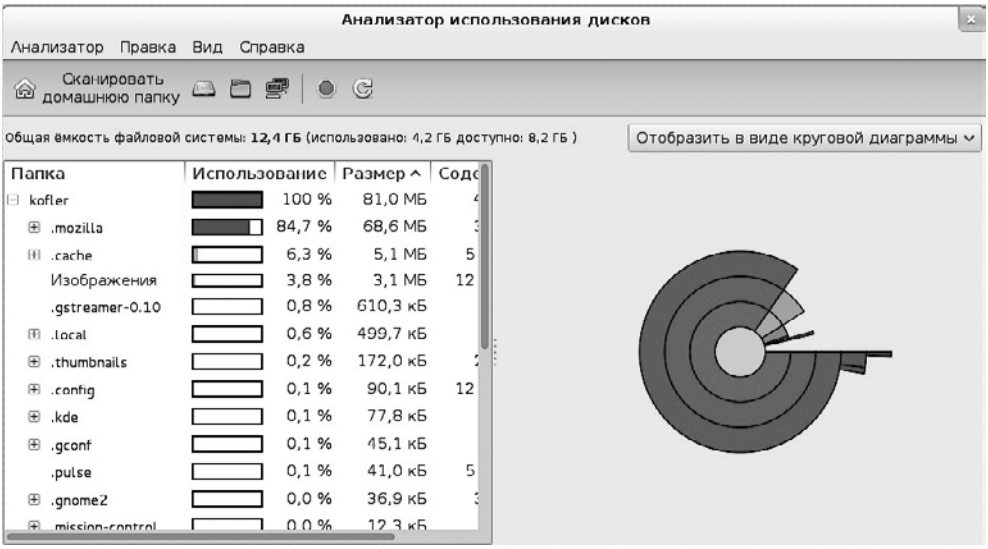


Рис. 2.4. Объем, занимаемый различными каталогами

Контакты (gnome-contacts). Программа Контакты позволяет управлять локальной адресной книгой или онлайновыми адресными книгами Google, Facebook и Windows Live. Конфигурация адресной книги задается при первом запуске программы или (впоследствии) в модуле системных настроек Онлайновые учетные записи (Online Accounts). Адресная информация может использоваться и другими программами Gnome.

Управление паролями и ключами (seahorse). В различных программах Gnome требуется вводить учетные данные пользователя и пароль. Чтобы каждой программе не приходилось самостоятельно отвечать за (максимально надежное) управление паролями, эта задача осуществляется в Gnome централизованно. Система защиты паролей используется, например, файловым менеджером, сетевым менеджером или почтовым клиентом Evolution. База данных паролей защищается основным паролем (master password), который запрашивается при первом включении программы. Все последующие операции доступа к базе данных возможны без дополнительного подтверждения.

Чтобы специально удалять отдельные записи из базы данных, запустите программу Пароли и ключи (`seahorse`). Кроме того, это приложение удобно для управления GPG- и SSH-ключами. Такие ключи нужны пользователям для шифрования или цифрового подписывания электронных сообщений, программистам — для подписывания кода, веб-разработчикам — для входа на удаленные машины и т. д.

Отображение файлов в форматах PDF и PostScript (программа Evince). При двойном щелчке на файле формата PDF или PS запускается программа Evince, которая отображает такой документ. Теперь можно просматривать документ, выводить на печать отдельные его страницы и т. д. (В случае работы со сложными PDF-документами, чтобы добиться более качественного отображения, лучше воспользоваться программой Adobe Reader.)

Окно терминала (gnome-terminal). За работу с окном терминала Gnome отвечает программа `gnome-terminal`. Она отличается некоторыми особенностями.

- Веб-адреса автоматически подчеркиваются, как только вы наводите на них указатель мыши. Правой кнопкой мыши запускается браузер, в котором открывается та или иная веб-страница.
- Когда вы перемещаете методом перетаскивания файлы или каталоги из файлового менеджера в командное окно (shell window), вставляется полное имя файла.
- С помощью сочетания клавиш Ctrl+«+» или Ctrl+«-» можно быстро изменять размер шрифта.
- Используя команду Файл ▸ Новая вкладка или сочетание клавиш Ctrl+Shift+T, можно открыть в окне новый терминал. Переключаться между терминалами можно щелчком кнопкой мыши либо сочетаниями клавиш Ctrl+Page up и Ctrl+Page down.
- Такие сочетания клавиш, как Alt+D, открывают меню окна терминала. Если вы предпочитаете использовать такие сочетания при работе со строкой ввода, выполните команду Правка ▸ Сочетания клавиш и установите флажок Деактивировать все сочетания клавиш при работе с меню.
- Множество других возможностей конфигурации дает команда Правка ▸ Профиль. Настройки можно присваивать различным профилям. После этого каждому терминалу можно присвоить собственный профиль.

Текстовый редактор (gedit). Стандартный текстовый редактор, применяемый в Gnome, называется `gedit`. Эту программу легко освоить, и она отлично подходит для решения простых задач.

Веб-браузер. В версии 3.6 официальный веб-браузер в Gnome называется не Epiphany, как ранее, а просто Web. Этот браузер очень хорошо интегрируется в рабочую настольную среду Gnome, даже притом, что многие пользователи предпочитают не его, а специализированные версии Firefox и Google Chrome. Но во многих дистрибутивах Web, он же Epiphany, по умолчанию не инсталлируется. Если вы хотите опробовать эту программу, установите пакет `epiphany` (в Debian и Ubuntu он называется `epiphany-browser`).

2.4. Конфигурация и внутреннее устройство

В Gnome 2.*n* большинство элементов Рабочего стола были конфигурируемыми: панели можно было прикреплять к любой из четырех сторон экрана, компоненты, находящиеся внутри панелей, можно было свободно выбирать и т. д. В Gnome 3.*n* эти вольности закончились! В системных настройках Gnome можно изменить обои Рабочего стола всего один раз. В качестве компенсации Gnome реализует новую концепцию расширяемости, которая позволяет устанавливать в браузере маленькие дополнительные программы, написанные на JavaScript. Это делается одним щелчком кнопкой мыши. Сначала с расширениями были сложности, но начиная с версии 3.4, эти компоненты (gnome extensions) работают превосходно. Такие компоненты

образуют базис для многочисленных полезных (и даже избыточных) дополнений и модификаций Gnome.

В этом разделе, наряду с системными настройками и расширениями Gnome, будут рассмотрены некоторые другие программы и вспомогательные средства, позволяющие оптимально сконфигурировать настольную рабочую среду Gnome. Именно здесь мы познакомимся и с некоторыми внутрисистемными особенностями Gnome.

Системные настройки

В системных настройках Gnome предоставляется целый арсенал инструментов для конфигурации Рабочего стола и самой системы (рис. 2.5). Быстрее всего эту программу можно запустить из меню Имя пользователя, расположенного в правом верхнем углу панели.

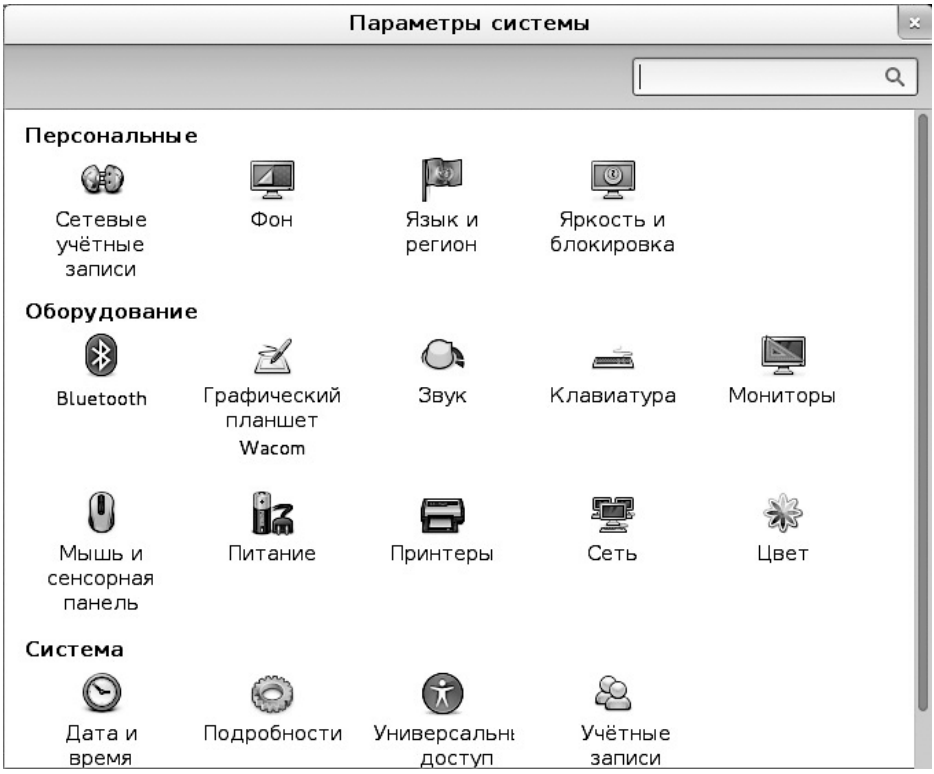


Рис. 2.5. Системные настройки Gnome

Модули системных настроек помогают не только задавать базовые настройки (такие как фон экрана, энергосберегающий режим, языковые настройки и т. д.), но и выполнять различные задачи, связанные с системным администрированием. К этой категории относится работа с сетевыми настройками, соединение по

Bluetooth, управление пользователями, настройки времени и т. д. Системные настройки в Gnome, как и другие инструменты, характеризуются крайним минимализмом. Например, с помощью модуля Пользовательские учетные записи можно, конечно, настраивать новые аккаунты, но управлять группами пользователей здесь уже нельзя. Если будет не хватать возможностей системных настроек Gnome, то можете воспользоваться консольными инструментами или поставляемыми вместе с системой конфигурационными программами вашего дистрибутива.

СОВЕТ

Не все инструменты, доступные в разделе Система, интегрированы в системные настройки. Другие системные инструменты и программы для настройки запускаются как обычные программы Действия ▸ Приложения ▸ Системные инструменты.

Принтер. В идеальном случае конфигурация принтера осуществляется автоматически. Многие дистрибутивы опознают большинство принтеров, подключаемых через USB, уже в момент соединения с устройством и автоматически выполняют конфигурацию. Всего через несколько секунд принтер готов к работе. Удобнее не бывает!

Если этот механизм не сработает либо если используется принтер, обслуживающий целую сеть, вам поможет новый модуль Принтер (Printer), появившийся в конфигурационном разделе. Во время тестирования мне удалось найти в системе один из двух моих сетевых принтеров, а второй — не удалось. Очевидно, конфигурационный инструмент может работать лишь с такими принтерами, которые он сам опознал в сети. И даже это у программы получалось не всегда — кстати, сетевой экран во время тестов был отключен. Указание хост-имени и IP-адреса принтера вручную не предусмотрено. О дивный новый мир Gnome!

Справиться с этой проблемой помогает программа `system-config-printer`, поставляемая с большинством дистрибутивов. Программа не новая и на вид не такая красивая, как модуль Принтер из системных настроек, но зато она проверена временем. Команда Add printer (Добавить принтер) открывает ассистент, который выстраивает список предлагаемых на выбор типов принтеров (в том числе, тот принтер, который может быть опознан в вашей системе). В случае с сетевыми принтерами, задача обычно решается с помощью App Socket ▸ HP Jet Direct (Сокет приложения ▸ HP Jet Direct). После выбора типа принтера вы указываете для него драйвер. При этом задается производитель и модель устройства. Все остальные настройки принтера (размер бумаги, дуплексный режим и т. д.) не являются обязательными и задаются в диалоговом окне Properties (Свойства).

Онлайновые аккаунты. В модуле Онлайновые аккаунты (Online Accounts) можно указать логин вашей учетной записи в Google или Windows Live. В перспективе планируется поддержка и других онлайновых служб. В дальнейшем аккаунты можно применять и в других приложениях Gnome, например в программе Контакты. Во время моих тестов интеграция онлайновых служб в Gnome функционировала, хотя и не слишком надежно.

Настройка программ по умолчанию. В большинстве дистрибутивов, основанных на Gnome, в качестве основного браузера применяется Firefox, в качестве почтового клиента — Evolution или Thunderbird и т. д. Если вы хотите, чтобы при

щелчке на соответствующих ссылках Gnome запускал другие программы, то соответствующие возможности настройки вы найдете в диалоговом окне системных параметров Подробности ▸ Приложения по умолчанию (рис. 2.6).

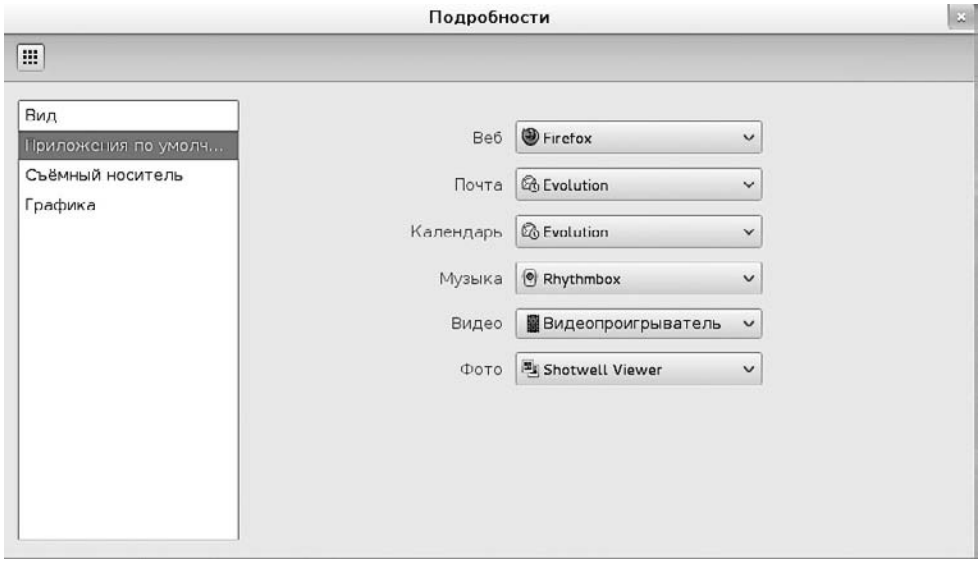


Рис. 2.6. Установка программ по умолчанию

О том, какую программу будет запускать Gnome при помещении в привод AudioCD или DVD с видео, а также при подключении к компьютеру MP3-плеера, можно узнать в разделе Съемный носитель.

Клавиатура. Настройки клавиатуры распределены на два модуля, входящих в состав системных настроек. В модуле Клавиатура определяются параметры повторного нажатия клавиши (key repeat), а также горячие клавиши. Настройки раскладки клавиатуры находятся модуле Регион и язык ▸ Раскладка.

Gnome Tweak Tool

С помощью отладочного инструмента Gnome (Gnome Tweak Tool, имя пакета обычно звучит как `gnome-tweak-tool`) можно изменять некоторые параметры Рабочего стола Gnome, настройка которых в официальном системном разделе не предусмотрена. Эта программа, в частности, позволяет указывать:

- какие кнопки будут представлены в панели инструментов окна;
- какой шрифт будет применяться для надписей на Рабочем столе;
- каково будет тематическое оформление окна;
- как будут вести себя ноутбуки с опущенным экраном;
- может ли файловый менеджер отображать на Рабочем столе ярлыки;
- должна ли в апплете Время отображаться также дата.

Некоторые изменения конфигурации вступают в силу лишь после перезапуска оболочки (Gnome Shell). Для этого нажмите Alt+F2+R+Enter. Другой вариант — просто выйти из системы и снова войти в нее.

Расширения оболочки Gnome

Работа командной оболочки Gnome в большой степени базируется на JavaScript. Поэтому всего нескольких строк на JavaScript достаточно, чтобы осуществлять значительные модификации Рабочего стола Gnome. В Gnome предусмотрен специальный механизм работы с расширениями. В версии 3.4 и выше такие расширения можно с легкостью скачивать в браузер, активизировать, а если что-то пойдет не так — то так же легко деактивировать. Следующий сайт — для тех, кому нравится все опробовать самостоятельно — https://extensions.gnome.org/.

Популярность тех или иных расширений одновременно показывает, где, по мнению пользователей Gnome, особенно назрела необходимость в улучшениях. Большинство расширений становятся активны сразу же после установки, лишь немногие требуют перезапуска оболочки Gnome (об этом см. выше) или повторного входа в систему. Официально сайт с расширениями для Gnome (на момент написания книги) еще является бета-версией. На мой взгляд, система уже функционирует превосходно. Определенную проблему представляет смена версий: если вы выполняете обновление на более свежей версии Gnome, то может оказаться, что некоторые расширения уже не функционируют.

Далее я расскажу о некоторых расширениях, с которыми мне доводилось работать в Gnome 3.4 и которые функционируют вполне нормально.

- Меню смены статуса (Alternate Status Menu). Это расширение позволяет включить в меню, расположенное с правого края панели, кроме записи Выход и записи Режим ожидания, Гибернация и Выключить.
- Смена вкладки (Alternate Tab). Если активировать данное расширение, то сочетание Alt+Tab будет функционировать как раньше.
- Меню AXE. Это расширение заменяет кнопку Действия на меню с широкими возможностями конфигурации, которое очень напоминает меню проекта Cinnamon из дистрибутива Linux Mint.
- Индикатор температуры процессора (CPU Temperature Indicator). Данное расширение отображает на панели инструментов температуру процессора. Данное расширение требует установки пакета `lm_sensors` или `lm-sensors`.
- Dash to Dock. Это расширение возвращает привычный вид док-панели Gnome, которая в новых версиях Gnome называется Dash. Обычная док-панель, в отличие от Dash, всегда видна на экране (за исключением случаев, когда она перекрывается открытым окном).
- Индикатор медиаплеера (Media Player Indicator). Данное расширение обеспечивает управление большинством медиаплееров (в частности, Banshee и Rhytmbox) через ярлык Панель, то есть как в Ubuntu.
- Panel Docklet и Window List. Оба этих расширения встраивают в панель инструментов список окон и упрощают таким образом переход между активными

окнами/программами. При этом Panel Docklet предлагает широкие возможности конфигурации.

- Remove Accessibility (Удаление доступности). При активации данного расширения с панели инструментов исчезает ярлык Для пользователей с ограниченными возможностями.
- System Monitor (Отслеживание системы). Это расширение встраивает в панель инструмент для контроля системных показателей и отображает, в частности, нагрузку на процессор, потребление памяти и т. п.

Конфигурационные файлы Gnome

База данных dconf. Параллельно с Gnome 3.0 была разработана новая система dconf, предназначенная для хранения программных настроек. Информация dconf находится в бинарном файле базы данных `.config/dconf/user`. Правда, не все программы Gnome используют при работе систему dconf.

В кругах разработчиков внедрение формата dconf вызвало противоречивые отзывы. Критики указывают, что в долгосрочной перспективе техническая поддержка такого двоичного файла чрезмерно усложнится — как это произошло с регистрационной базой данных в Windows. Кроме того, представляется рискованным хранить настройки многочисленных программ в единственном двоичном файле. Если эти файлы будут стерты по ошибке или из-за сбоя в системе, то это затронет значительные части настольной системы.

Преимущество dconf по сравнению с более старой системой gconf (о ней рассказано ниже) заключается в значительном повышении скорости доступа. Во время запуска Gnome системе требуется считывать многочисленные настройки. При полном переходе на систему dconf запуск Gnome значительно ускоряется.

Программы Gnome обращаются к базе данных dconf непосредственно через функции API (интерфейса программирования приложений). Если вы хотите считать или изменить настройки dconf извне, то установите пакет `dconf-tools` и запустите программу `dconf-editor`. В левой части пользовательского интерфейса этой программы отображается древовидная структура всех каталогов с настройками, а справа перечисляются параметры, содержащиеся в текущем каталоге.

Команда `gsettings` позволяет изменять настройки `dconf` в терминале или с помощью специального скрипта. Благодаря следующей команде Nautilus не требует подтверждения при удалении файлов:

```
user$ gsettings set org.gnome.nautilus.preferences confirm-trash false
```

База данных gconf. Сравнительно старые программы Gnome, а также те, которые не были адаптированы для работы с системой dconf, обычно сохраняют свои настройки в базе данных gconf. Внутренняя структура этой базы данных представляет собой множество мелких XML-файлов, которые сохраняются в каталоге `.gconf`, а также в его подкаталогах. Для изменения настроек gconf также существует специальная программа `gconf-editor` с простым пользовательским интерфейсом. Она очень напоминает по виду программу `dconf-editor`, но в отличие от нее располагает поисковой функцией.

Оконные кнопки. Изменять настройки можно не только с помощью пользовательского интерфейса, но и используя команду `gconftool-2`. Следующая команда открывает окно с двумя дополнительными кнопками — Свернуть и Развернуть:

```
user$ gconftool-2 --set /desktop/gnome/shell/windows/button_layout \
      --type string ":minimize,maximize,close"
```

Если вы хотите расположить оконные кнопки у левого края (как в Mac OS X или в Ubuntu), то выполните следующую команду:

```
user$ gconftool-2 --set /desktop/gnome/shell/windows/button_layout \
      --type string "close,minimize,maximize"
```

Чтобы измененные настройки вступили в силу, нужно перезапустить оболочку Gnome. Для этого нажмите Alt+F2+R+Enter.

Внутреннее устройство системы

«За кулисами» управлением окнами и запуском программ (перспектива Действия) занимается программа Gnome Shell (Командная оболочка Gnome). Эта программа, в свою очередь, интегрирована в новый менеджер окон, который называется Mutter. Mutter — это следующее поколение оконного менеджера Metacity. Важнейшее из нововведений — это поддержка 3D-эффектов. Поэтому для работы с Gnome 3.*n* программа Compiz больше не требуется.

Автозапуск. При старте Gnome автоматически запускается множество более мелких программ. То, какие именно приложения будут запускаться автоматически, определяют файлы `*.desktop`. Они хранятся в следующих каталогах автозапуска:

- `~/.config/autostart/*.desktop` — личные программы, запускаемые автоматически;
- `/usr/share/gnome/autostart/*.desktop` — глобальные настройки автозапуска приложений в Gnome;
- `/etc/xdg/autostart/*.desktop` — глобальные настройки автозапуска программ во всех настольных системах, то есть в Gnome и KDE.

В Gnome 3.*n*, к сожалению, уже нет графического пользовательского интерфейса для управления автозапуском программ. Иногда файлы `*.desktop` приходится удалять или создавать вручную. Организация подобного файла понятна из приведенного ниже примера. Файл предназначен для воспроизведения звуковых сигналов при входе в систему:

```
[Desktop Entry]
Type=Application
Name=GNOME Login Sound
Comment=Plays a sound whenever you log in
Exec=/usr/bin/canberra-gtk-play --id="desktop-login" --description="GNOME Login"
OnlyShowIn=GNOME;
AutostartCondition=GSettings org.gnome.desktop.sound event-sounds
X-GNOME-Autostart-Phase=Application
X-GNOME-Provides=login-sound
```

Дисплейный менеджер Gnome. В дистрибутивах, работающих с Gnome, за вход в систему отвечает, как правило, дисплейный менеджер Gnome (gdm). Исключением является Ubuntu, где в версии 11.10 и выше применяется «Легкий дисплейный менеджер» (`lightdm`).

MIME. При двойном щелчке на MP3-файле в Nautilus автоматически открывается программа Rhytmbox или Banshee. Это происходит благодаря действующим в Gnome настройкам MIME-типов. Аббревиатура MIME означает Multipurpose Internet Mail Extensions (многоцелевые расширения интернет-почты). Это своего рода база данных, обеспечивающая соотнесение файлов конкретных типов с теми или иными программами.

Проще всего вносить изменения в конфигурацию MIME-типов непосредственно в файловом менеджере. Там вы щелкаете на интересующем вас файле, выполняете в контекстном меню команду Свойства ▸ Открыть с помощью и выбираете желаемую программу. В дальнейшем данная настройка будет действовать для всех файлов с конкретным расширением.

Индивидуальные изменения конфигурации MIME сохраняются здесь:

```
~/.local/share/mime/*
~/.local/share/applications/mimeapps.list
```

Каталоги и скрипты XDG

Несколько лет назад в рамках проекта Portland было разработано несколько общих стандартов. Они помогают правильно интегрировать программы в настольную среду, независимо от Gnome и KDE. Позже группа X Desktop Group (XDG) продолжила эти разработки, их результатом стал проект freedesktop.org.

Стандартные каталоги. При первом входе в систему в домашнем каталоге создаются подкаталоги Images (Изображения), Documents (Документы), Music (Музыка), Common (Общие), Videos (Видео) и Templates (Шаблоны). Названия каталогов выбираются в зависимости от языковых настроек. На внутрисистемном уровне за управление каталогами отвечает пакет `xdg-user-dirs`.

Конфигурация осуществляется в файле `~/user-dirs.dirs`. Этот каталог гарантирует, что программы, совместимые со стандартами XDG, находят нужные каталоги независимо от языковых настроек. В случае изменения языковых настроек в Gnomy каталоги переименовываются соответствующим образом после запроса о подтверждении (пакет `xdg-user-dirs-gtk`).

Если стандартные каталоги вас не устраивают, то удалите их и создайте вот такой новый файл:

```
# ~/.config/user-dirs.conf
enabled=False
```

Эту настройку можно задействовать и на уровне системы, в файле `/etc/xdg/user-dirs.conf`.

Конфигурационные каталоги. Многие, если (к сожалению) не все, программы Gnome и KDE используют для хранения конфигурационных настроек и внутренних данных специально предназначенные для этого каталоги. Имена этих катало-

гов начинаются с точки, следовательно, данные каталоги являются скрытыми. Соответственно, такие каталоги по умолчанию не отображаются в файловом менеджере.

- Каталог `.cache` предназначен для хранения временных файлов, которые по мере необходимости можно создавать заново. Здесь находятся, например, миниатюры (thumbnails), поисковые индексы и т. д. Благодаря буферизации, ускоряются наиболее частотные рабочие процессы.
- Каталог `.config` предназначен для хранения программных настроек, причем каждая программа применяет собственный подкаталог.
- В каталоге `.local` сохраняются пользовательские данные. Обычно программа создает для этой цели подкаталог `share/programmname`.

XDG-скрипты. Пакет `xdg-utils` содержит следующие скрипты (подробное описание выводится командой `man xdg`):

- `xdg-desktop-menu` — добавляет в меню Рабочего стола новую запись;
- `xdg-desktop-icon` — устанавливает на Рабочем столе новый ярлык;
- `xdg-icon-resource` — устанавливает ресурсы ярлыков;
- `xdg-mime` — запрашивает базу данных MIME или создает новый MIME-тип;
- `xdg-open` — открывает документ в стандартной программе, заданной пользователем;
- `xdg-email` — отправляет электронное сообщение с помощью стандартного клиента, заданного пользователем;
- `xdg-screensaver` — управляет скринсейвером.

2.5. Варианты Gnome

После выхода Gnome 3 мир Gnome стал довольно противоречивым. Некоторые дистрибутивы по-прежнему используют Gnome 2 (таковы, например, Debian 6 и RHEL 6) либо ответвление Gnome 2 под названием MATE (Linux Mint MATE). Другие дистрибутивы, хотя и применяют основные компоненты Gnome 3, модифицируют функционал и внешний вид Рабочего стола Gnome. Иногда вносятся настолько значительные изменения, что такие системы уже сложно назвать Gnome 3 (например, к таким системам относится Linux Mint Cinnamon). В этом разделе дается краткий обзор важнейших вариантов Gnome, за исключением Unity.

Gnome 2. В сравнительно старых дистрибутивах по-прежнему широко применяется Gnome 2. Основным характерным признаком Gnome 2 являются две панели. В верхней панели обычно находится стартовое меню, а также некоторые апплеты. Внизу располагается панель задач. Для внесения новых записей в стартовое меню щелкните правой кнопкой мыши и выполните команду Add this launcher to panel» (Добавить на панель для быстрого запуска). Правой кнопкой мыши также можно добавлять элементы на панель, перемещать их и удалять. Если вы хотите сэкономить место, можно удалить панель и разместить находившиеся в ней апплеты в другой панели.

Резервный режим для Gnome 3. Для работы с Gnome 3 требуется графическая карта с поддержкой 3D-графики, а также соответствующий драйвер. Если аппаратные требования не выполняются, то активируется так называемый резервный режим. При этом Gnome 3 начинает выглядеть примерно как Gnome 2 (рис. 2.7). В любом случае апплеты Gnome 2 в резервном режиме не работают.

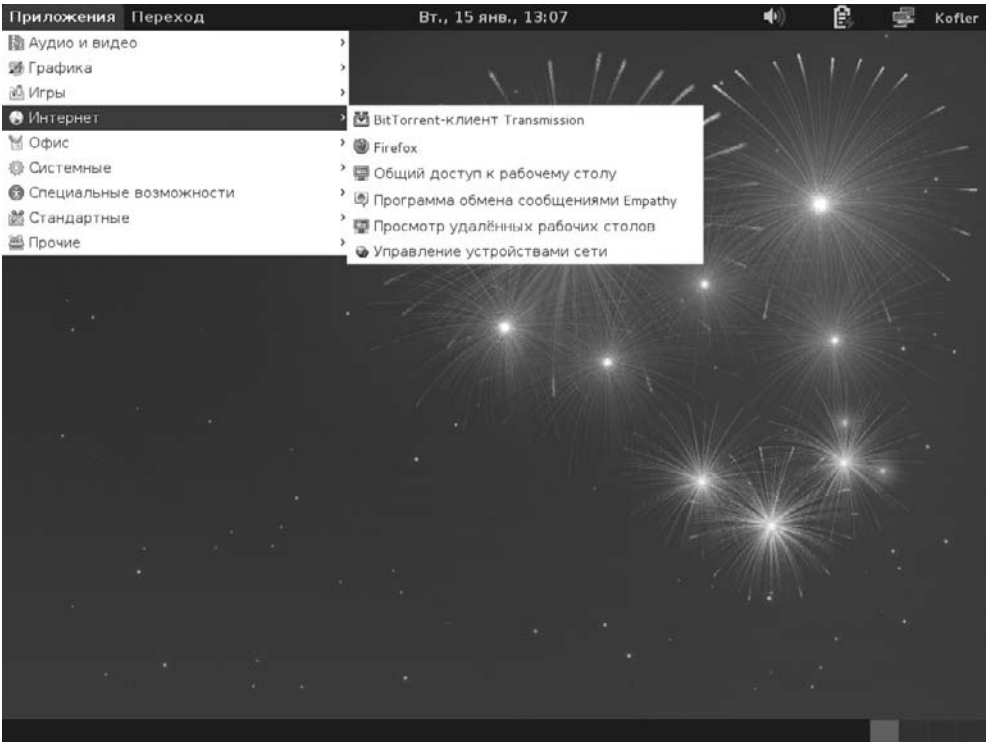


Рис. 2.7. Резервный режим Gnome 3 по виду напоминает Gnome 2

Существенное отличие такого режима от Gnome 2 заключается в том, что при внесении любых изменений на панель требуется нажимать не только левую или правую кнопки мыши, но и клавишу Alt. Таким образом Рабочий стол предохраняется от нежелательных изменений.

Вероятно, в будущих версиях Gnome резервный режим исчезнет. Fedora 17 демонстрирует, что вполне возможно обеспечить достаточно качественную поддержку Gnome 3 и без функций трехмерной графики на графической карте. Недостающие графические функции эмулируются процессором, работа в большинстве случаев протекает достаточно быстро. За эмуляцию отвечает библиотека `llvmpipe`. Если конфигурировать соответствующим образом и другие дистрибутивы, то резервный режим становится излишним.

MATE. С момента выхода Gnome 3 Gnome 2 больше не поддерживается. Таким образом, даже те дистрибутивы, которые вполне обходились Gnome 2, были бы вынуждены рано или поздно перейти на Gnome 3 — если бы не MATE.

MATE — это ответвление Gnome 2 (http://mate-desktop.org). Соответственно, в проекте MATE используется код Gnome 2, некоторые компоненты получили новые названия (во избежание конфликтов с проектом Gnome). В дальнейшем идет лишь работа над ошибками. Серьезные изменения или обновления MATE не планируются.

В том, как красиво может выглядеть Рабочий стол на базе Gnome 2 (то есть MATE), можно убедиться на примере дистрибутива Linux Mint 13 Maya. В стандартной конфигурации присутствует всего одна панель с индивидуально оформляемым главным меню. Это меню построено на базе апплета Main Menu, который применялся и в сравнительно старых версиях openSUSE.

Cinnamon. Рабочий стол Cinnamon — это дополнение для Gnome 3, созданное разработчиками Linux Mint. Cinnamon пытается сконфигурировать Gnome 3 таким образом, чтобы работа с ним не отличалась от Gnome 2, то есть присутствовали все привычные панели, не было док-панели и т. д. Собственно говоря, получилось на удивление хорошо. Но еще нет уверенности, что у этого проекта большое будущее: ведь пользователи Cinnamon лишены многих нововведений, появившихся в Gnome 3, кроме того, потеряна совместимость с Gnome 2. Cinnamon использует собственные апплеты и расширения, несовместимые с Gnome 2, Gnome 3 и Unity. И все же на сайте Cinnamon для свободной загрузки выложено удивительно много апплетов и расширений. Подробнее о Cinnamon рассказано здесь: http://cinnamon.linuxmint.com/.

3 KDE

KDE является популярной альтернативой настольной среды Gnome, которая была описана в предыдущей главе. В принципе, KDE решает те же задачи, что и Gnome, но выглядит иначе и внутрисистемно основывается на других библиотеках и протоколах. Сокращение KDE первоначально расшифровывалось как Kool Desktop Environment (Классная настольная среда), позже стали говорить просто K Desktop Environment. KDE основана на Qt — свободно распространяемой библиотеке, разработку которой начала компания Troll Tech. Позже работу над библиотекой продолжили в Nokia и в Digia. Подробная информация по KDE приводится на сайте http://kde.org/.

По сравнению с Gnome, в KDE содержится больше таких специальных функций и возможностей конфигурации, которые нравятся опытным пользователям Linux. Но в то же время система немного неудобна в использовании. Именно поэтому многие дистрибутивы по умолчанию работают с Gnome.

Если в вашем дистрибутиве предоставляются соответствующие пакеты, то ничего не мешает параллельно установить Gnome и KDE. В таком случае при входе в систему вы сможете указать, с какой системой собираетесь работать — Gnome или KDE (обычно для этого предназначается кнопка или пункт меню Session (Сессия)).

Основной темой данной главы являются базовые функции KDE. Разумеется, в различных дистрибутивах KDE может выглядеть совершенно по-разному. Это касается и экрана для входа в систему, и записей в меню Пуск, и оформления Рабочего стола, а также выбора поставляемых вместе с системой программ и соответствующих возможностей конфигурации. Эта глава была написана на основе KDE 4.9, данную систему я протестировал на бета-версии Kubuntu 12.10.

KDE и работа с мышью. Пожалуй, самая очевидная разница между KDE и другими пользовательскими интерфейсами заключается в работе с мышью. Чтобы открыть файл и выполнить другие подобные операции, в KDE достаточно одного щелчка кнопкой мыши (а не двух). К этому поначалу нужно привыкнуть, но эта особенность делает работу эффективной и удобной. Если вы не хотите переучиваться, то можно просто настроить в KDE двойной щелчок. Для этого откройте в меню KDE системные настройки (System Settings), перейдите в модуль Input Devices (Устройства ввода) и установите флажок Double-click to open files and folders (Открывать файлы и каталоги двойным щелчком).

3.1. Организация Рабочего стола

Вход в систему и выход из нее. Прежде чем приступать к работе с KDE, нужно войти в систему под вашим именем пользователя (логином) и паролем. Если кроме KDE вы установили и другие системы Рабочего стола либо оконный менеджер, в меню можно выбрать желаемую настольную систему.

Чтобы выйти из системы и перезагрузить компьютер, выберите в меню KDE запись Quit ▸ Log Out (Закончить работу ▸ Выйти из системы) или Quit ▸ Reboot (Закончить работу ▸ Перезагрузка). При наличии соответствующего аппаратного обеспечения на данном этапе вы можете перевести компьютер в состояние гибернации или в энергосберегающий режим.

Смена пользователя. В меню KDE присутствует команда Quit ▸ Change User (Закончить работу ▸ Сменить пользователя). Благодаря ей, если с компьютером должен поработать другой пользователь, первый пользователь может не завершать все свои программы. Внутри системы для каждого пользователя запускается собственная графическая система (X-сервер). Поэтому при нескольких параллельных входах в систему потребляется очень много ресурсов, и такие параллельные входы нормально работают лишь на очень быстрых компьютерах. Для быстрого переключения между пользователями в большинстве дистрибутивов применяются нижеперечисленные сочетания клавиш (лишь в Fedora 17 действует сочетание Ctrl+Alt+F1):

- Ctrl+Alt+F7 — первый пользователь;
- Ctrl+Alt+F8 — второй пользователь;
- Ctrl+Alt+F9 — третий пользователь.

Рабочий стол. На рис. 3.1 показан Рабочий стол системы KDE 4.9. Как уже говорилось выше, внешний вид Рабочего стола сильно зависит от дистрибутива и от конфигурации всей системы. По умолчанию Рабочий стол обычно состоит из одной панели, идущей по нижнему краю экрана, и из самой рабочей области. На панели содержится меню KDE, здесь же могут находиться ярлыки, предназначенные для быстрого запуска определенных программ, панель задач с пиктограммами всех открытых окон, а также различные вспомогательные программы.

Plasma. Plasma — это, пожалуй, важнейший объект KDE. Он позволяет выкладывать на Рабочий стол или на панель интерактивные объекты и работать с ними. Эта функция сравнима с панелью инструментов (Dashboard) в Apple.

Плазмоиды. Сама рабочая область (Рабочий стол) изначально почти пуста. Мини-программы можно выполнять прямо на Рабочем столе или на панели, в номенклатуре KDE такие программки называются «плазмоиды». С помощью команды Add Miniprogramms (Добавить мини-программы) из контекстного меню или кнопки Toolbox (Инструментарий), расположенной в верхнем левом углу экрана, можно добавлять плазмоиды на Рабочий стол (рис. 3.2).

К сожалению, плазмоиды организованы в виде очень узкого списка и выбирать их довольно неудобно. Тем не менее, плазмоиды можно сортировать по категориям.

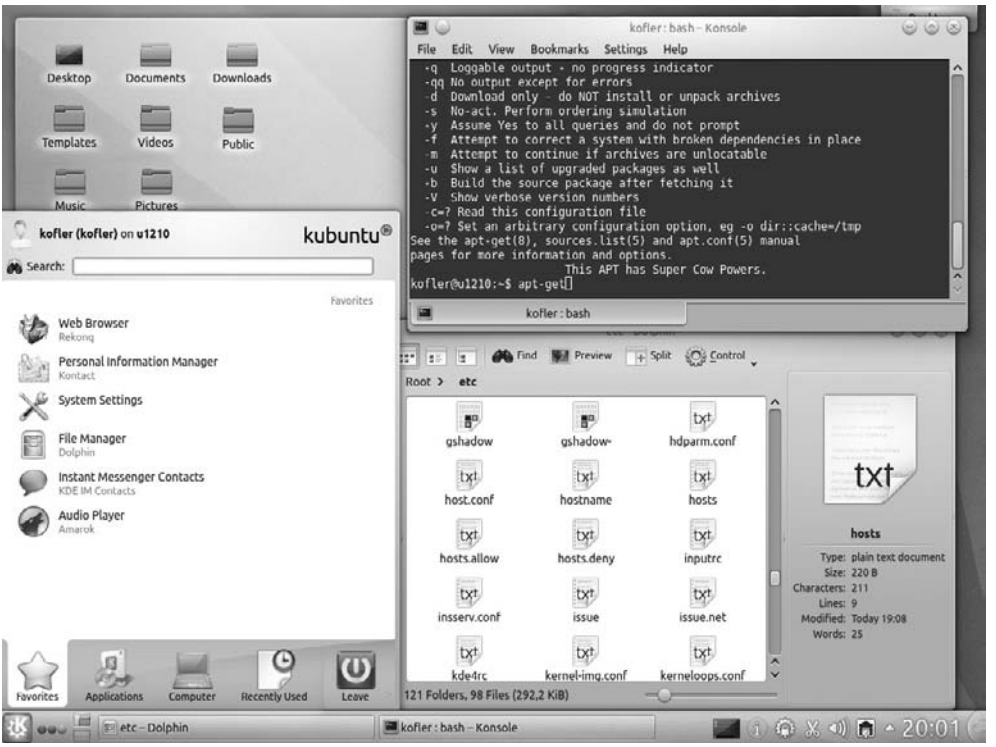


Рис. 3.1. Рабочий стол KDE

Рис. 3.2. Вставка мини-программ (плазмоидов)

На Рабочем столе можно выкладывать ярлыки, а их оптическим представлением будет заниматься Plasma. Но сейчас рекомендуется использовать плазмоид Folder View (Перспектива «Каталоги»). Таким образом, можно отображать каталог `Desktop` и содержащиеся в нем ярлыки в виде своеобразного Plasma-окна (см. на рис. 3.1, *слева*). При этом речь не идет об окне в привычном смысле этого слова. Плазмоид Folder View (Перспектива «Каталоги») всегда находится за всеми обычными окнами и выглядит немного иначе.

СОВЕТ

Лично я — не любитель значков, мини-программ и прочих настольных объектов. Когда я работаю, почти вся рабочая область на экране закрыта несколькими большими окнами. Если вам нравится работать с ярлыками и плазмоидами, обратите внимание на сочетание клавиш Alt+F12: оно поднимает элементы Рабочего стола на передний план и отображает все окна затемненными

на заднем плане. Если еще раз нажать Alt+F12 либо Esc, Рабочий стол вернется в прежнее состояние.

При наведении указателя мыши на ярлык или плазмоид сбоку от этого элемента появляется несколько кнопок, с помощью которых можно изменять размер и конфигурацию мини-программы. Эти настройки сохраняются в файле `~/.kde4/share/config/plasma-desktop-appletsrc`.

Панели. Панель — это прямоугольная область, которая располагается у края экрана (обычно у нижнего). Сама по себе панель не выполняет никаких функций, а играет роль контейнера для мини-программ. Даже такие базовые элементы, как меню и линейка задач реализованы в KDE 4 в виде плазмоидов! Поэтому в принципе возможно (хотя, это и не распространенная практика) обходиться вообще без панели и размещать меню, линейку задач и остальные подобные элементы прямо на Рабочем столе. Важнейшее достоинство панели заключается в том, что окна не перекрывают эту область экрана. Кроме того, при компактном размещении нескольких плазмоидов на одной панели экономится много места.

С помощью команды контекстного меню Control Panel Settings (Настройки контрольной панели) можно изменять размер, положение и другие свойства панели, а также добавлять на нее мини-программы, перемещать их и удалять. Для этого сверху или рядом с панелью выводится меню. Цвет или фоновый рисунок панели обычно определяется дизайном Рабочего стола. Изменить это изображение можно, лишь выбрав другой дизайн.

Важные мини-программы (плазмоиды)

Меню KDE (быстрый старт). Пожалуй, важнейшим плазмоидом является меню KDE для быстрого старта (рис. 3.3). Оно делится на пять разделов.

- Favorites (Избранное) — содержит важнейшие программы по выбору пользователя. Панель избранного не создается автоматически (например, в зависимости от частоты использования тех или иных программ). Напротив, выбирая программы для области избранного, их нужно явно добавлять туда из других категорий меню с помощью специальной команды контекстного меню.
- Applications (Приложения) — включает в себя иерархически выстроенный перечень всех программ. Листание между программами отличается от обычного выбора из меню — к такому листанию требуется привыкнуть. С одной стороны, удобно, что не приходится все время удерживать кнопку мыши. С другой — щелкать приходится гораздо чаще, чем ранее.
- Computer (Компьютер) — позволяет запускать различные административные программы, а также открывать важные каталоги: личный каталог, основной каталог, в зависимости от конфигурации различные внешние носители данных, корзину и т. д.
- Recently Used (История) — содержит список программ, которые недавно запускались, а также последние использовавшиеся файлы и каталоги
- Leave (Выход) — включает в себя команды для выхода из системы, смены пользователя или перезагрузки компьютера.

Рис. 3.3. Меню KDE

Кроме категорий, в меню KDE также содержится поисковая функция. Она очень удобна для быстрого запуска программ, так как позволяет открывать их без утомительного поиска в закладках меню Programs (Программы). Вы можете сами модифицировать меню. Для этого нажмите кнопку запуска меню правой кнопкой мыши и откройте редактор меню (Menu Editor). Востребованные программы можно перетащить в свободную часть панели или Рабочего стола. Там появятся соответствующие ярлыки, и запускать эти программы не составит труда.

Если вы больше привыкли использовать обычное меню KDE 3, можно продолжать работать с ним. Для этого щелкните на кнопке запуска меню правой кнопкой мыши и выполните команду Switch to classic menu style (Перейти к классическому оформлению меню).

Линейка задач (линейка окон). Мини-программа Window Bar (Линейка окон) отображает пиктограмму каждого из открытых окон и, соответственно, очень напоминает знакомую Панель задач Windows. В диалоговом окне настроек можно указать, следует ли объединять в одну группу несколько открытых окон одной и той же программы (например, окна Gimp). Здесь же задается, как следует сортировать окна (по алфавиту, в порядке запуска программ или вручную).

Линейку окон можно сравнить также с док-панелью, используемой в Mac OS X и Windows 7. На ней, как и на док-панели, можно выкладывать ярлыки для запуска программ, не работающих в настоящий момент. Для этого выполните в работающей программе команду контекстного меню Extended ▸ Show when not running (Дополнительно ▸ Отображать в неактивном состоянии).

Рабочие области. Рабочие области позволяют распределять окна работающих программ на несколько виртуальных Рабочих столов и переключаться между этими Рабочими столами. Это упрощает работу и повышает наглядность, если вы одновременно открываете несколько окон. За управление рабочими областями отвечает плазмоид Workspace Switcher (Переключатель рабочих областей). В меню его настроек вы указываете желаемое количество рабочих областей, а также различные другие параметры.

Если какие-то окна постоянно нужны вам под рукой, то вы можете обозначить их так, чтобы они отображались не в одной, а сразу во всех рабочих областях. Для этого откройте мышью или сочетанием клавиш Alt+Пробел меню окна и выполните команду On Workspace ▸ All Workspaces (В рабочей области ▸ Все рабочие области).

Действия. Действия (Activities) продолжают ту же идею, что и рабочие области. Кнопка действий, обозначенная тремя разноцветными точками, позволяет переходить между разными Рабочими столами. При этом действие запускается кнопкой со стрелкой, а черная квадратная кнопка завершает действие.

Но действия — это не просто заново реализованные рабочие области. Каждое действие может иметь собственный фон Рабочего стола, выполнять собственные плазмоиды и управлять собственными рабочими областями. При первом применении действия могут автоматически запускать программы (я не нашел, как настраивать эту характеристику). Некоторые действия определены заранее, к ним относятся Desktop ((Рабочий стол); применяется по умолчанию), Search and Execute (Поиск и исполнение) и Desktop Icons (Ярлыки Рабочего стола). Вы можете сами определять и создавать новые действия.

Хотя концепция действий довольно интересна, их эффективность значительно снижается из-за недостатка документации и плохо структурированной настройки.

Системный раздел. Если панель содержит так называемый системный раздел, ваше внимание могут привлекать находящиеся в панели фоновые программы (например, если доступны новые обновления или вы получили новое электронное письмо). Этот системный раздел обычно находится в правом нижнем углу панели. Он не выполняет никаких функций, а просто занимает место, в котором можно размещать ярлыки других программ. Вам доведется обращать внимание на системный раздел лишь в тех случаях, когда на фоне панели по какой-то причине появятся уведомления об обновлениях или о других событиях.

Наблюдение за устройствами. Плазмоид Device Monitoring (Наблюдение за устройствами) сообщает вам о новых подключенных носителях информации и помогает открывать их файловые системы либо вновь правильно отключать эти устройства от дерева каталогов (`unmount`).

Быстрый доступ. Мини-программа Quick Access (Быстрый доступ) обеспечивает быстрый доступ к содержимому домашнего каталога и всех находящихся в нем подкаталогов и файлов. Щелкнув на файле, вы запускаете обслуживающую его программу. Для просмотра отдельных каталогов в программе Dolphin щелкните правой кнопкой мыши и выберите в контекстном меню команду Open (Открыть).

Управление окнами

Большинство применяемых в KDE функций, связанных с управлением окнами, определенно знакомы вам из опыта работы с другими операционными системами для настольных компьютеров. Окна можно перемещать, разворачивать, сворачивать и т. д. Кроме того, в KDE есть некоторые специфические особенности, которые на первый взгляд могут быть неочевидны.

- **Разворачивание окон по вертикали/горизонтали.** Если щелкнуть на кнопке для развертывания окна средней или правой кнопкой мыши, то окно увеличится, соответственно, только по вертикали или только по горизонтали.
- **Группирование окон.** Чтобы объединить в группу несколько окон, в которых содержится похожая информация, щелкните правой кнопкой мыши на титульной панели окна и выполните команду Move window to group ▸ Window name (Добавить окно в группу ▸ Имя окна). Тогда актуальное окно будет добавлено в окно, называющееся Window Name (Имя окна). Обе программы станут диалоговыми вкладками одного и того же окна (рис. 3.4). Эти окна можно будет вместе сворачивать, разворачивать и перемещать. Вращая колесико мыши на титульной панели сегментированного окна, вы переходите в следующую вкладку. С помощью команды Remove window from group (Удалить окно из группы) вы снова можете превратить диалоговую вкладку в самостоятельное окно.

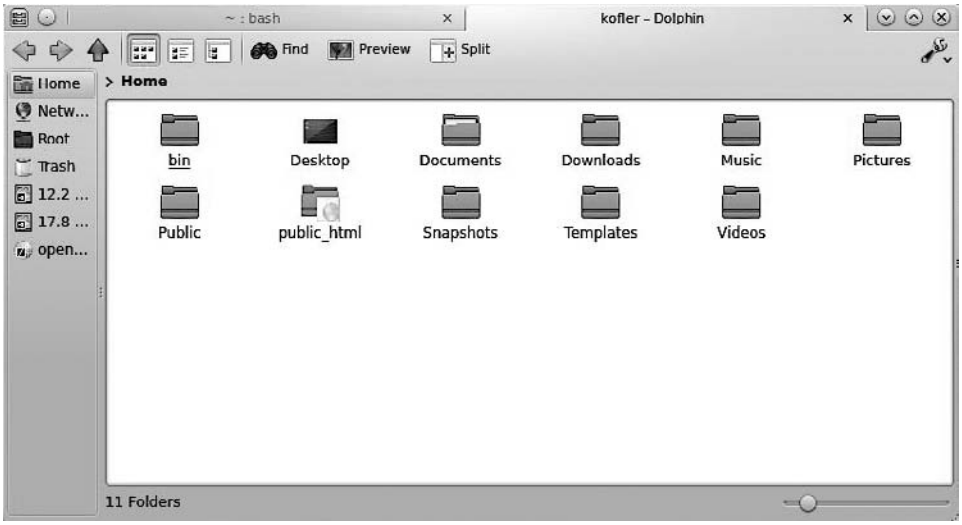


Рис. 3.4. Группа из двух окон, в которую входит окно терминала и окно программы Dolphin

- **Размещение окон в правой или левой половине.** Если вы перемещаете окно с помощью мыши и при этом продвигаете указатель мыши до левого или до правого края монитора, то окно размещается так, что оно заполняет правую или левую половину экрана либо четверть экрана. Это очень удобно, в первую очередь, на широкоэкранных мониторах.

- **Мозаичный режим.** Нажав сочетание клавиш Shift+Alt+F11, вы переходите в мозаичный режим (tiling mode). Теперь все активные окна распределяются по экрану, не накладываясь друг на друга. Активное окно занимает правую половину экрана, остальные занимают правую верхнюю четверть, и т. д. Если вы открываете новые окна, сворачиваете окна, закрываете их или перемещаете, то расположение окон, как правило (но не всегда) обновляется.

  К работе в мозаичном режиме требуется привыкнуть. Если такой режим и целесообразен, то только, пожалуй, на очень большом мониторе. Различные параметры для управления этим режимом находятся в системном модуле Window Behavior (Поведение окон). Переход в этот модуль осуществляется в диалоговом окне Window Behavior ▸ Extended (Поведение окон ▸ Дополнительно). Чтобы деактивировать этот режим, нужно еще раз нажать Shift+Alt+F11.

3.2. Dolphin

В KDE 4 программа Dolphin является основным файловым менеджером, заменившим универсальную программу Konquero». Dolphin, правда, нельзя использовать в качестве веб-браузера, зато, по сравнению с Konqueror, он обладает значительно более наглядным пользовательским интерфейсом. Разумеется, любители Konqueror могут и далее работать с этой программой (подробнее об этом — в следующем разделе).

Перспективы (Views). Dolphin запускается в меню KDE командой Favorites ▸ File Manager (Избранное ▸ Файловый менеджер) или с помощью команды Computer ▸ Personal Folder (Компьютер ▸ Личный каталог). Основные функции программы почти не требуют объяснений: в центре окна показываются файлы, причем есть три режима отображения: Symbols (Символы), Details (Таблица) и Compact (Компактный режим). Они активизируются, соответственно, сочетаниями клавиш от Ctrl+1 до Ctrl+3.

Очень удобна при этом функция группирования: Settings ▸ Group Items (Настройки ▸ Сгруппировать элементы). В таком случае файлы можно объединять по общему типу или по начальным буквам.

Кнопка Preview (Предварительный просмотр) позволяет независимо от режима отображения, открывать в режиме предварительного просмотра изображения и документы. Размер изображений в режиме предварительного просмотра можно настраивать с помощью ползунка. По умолчанию Dolphin создает экземпляр для предварительного просмотра лишь при условии, что размер просматриваемого файла не превышает 3 Мбайт. Вы можете увеличить это предельное значение командой Settings ▸ Adjust Dolphin (Настройки ▸ Настроить Dolphin). Это делается в диалоговом окне General ▸ Preview (Общие ▸ Предварительный просмотр). На данном этапе вы также можете указать, для файлов каких типов будет выполняться предварительный просмотр.

Для перемещения и копирования элементов вы можете делить внутреннюю область программы с помощью команды Share View (Разделить вид) по горизонтали или по вертикали. Так можно отображать в одном окне два каталога.

Актуальный каталог показывается на навигационной панели под меню. Сочетание клавиш Ctrl+L позволяет переключаться между двумя вариантами просмотра в этой панели. Отдельные каталоги отображаются либо в виде кнопок (в таком случае можно быстро переключаться между каталогами), либо в текстовой форме — так можно быстро ввести новый каталог.

Независимо от того, какой вариант отображения активен в настоящий момент, вы можете добавить новый каталог, нажав клавишу F6 на клавиатуре.

Боковые панели. Левее, правее и ниже содержимого окна можно отобразить терминал, иерархию каталогов, список часто используемых мест, а также дополнительную информацию. Это делается с помощью команды View ▸ Sidebars (Вид ▸ Боковые панели), а также клавишами F4, F7, F9 и F11. В списке мест можно методом перетаскивания размещать новые каталоги.

Меню. В современных версиях KDE программа Dolphin по умолчанию отображается без меню. Доступ к, казалось бы, отсутствующим командам, скрывается под кнопкой Settings (Настройки). Если вам больше нравится традиционное меню, его можно включать и отключать сочетанием клавиш Ctrl+M.

Выделение файлов. На выделении файлов следует остановиться особо. При стандартных настройках KDE одного щелчка для этого недостаточно, поскольку после него файл отображается или выполняется. Поэтому вместе с кнопкой мыши нужно нажимать Ctrl (при многократном выделении) или Shift (при выборе диапазона).

В системе предлагается и еще более интересный режим выделения. Если некоторое время подержать указатель мыши на файле или каталоге (такое действие называется «наведение» или по-английски hover), то рядом со стрелкой появляется зеленый плюсик. Щелкнув на этом символе, вы выделите файл. Если файл уже выделен, то в данном случае рядом со стрелкой мыши появляется красный знак «минус», с помощью которого можно снова снять выделение.

Поиск файлов. С помощью кнопки Search (Поиск) можно искать файлы. В любом случае эта поисковая функция настолько же медленна, насколько и проста. Если вам требуется более эффективная функция настольного поиска, то в системных настройках модуля Desktop Search активизируйте программу индексирования файлов Nepomuk. Но, по моему опыту, настольный поиск в KDE пока организован плохо (потребляется много ресурсов, выдаваемые поисковые результаты слишком объемны).

Удаление файлов. Если вы удаляете файлы и каталоги, то сначала они попадают в корзину. Чтобы просмотреть содержимое корзины, щелкните на соответствующей записи на боковой панели Places (Места). Эта панель открывается клавишей F9. Только после того как вы отметите все нужные вам объекты и нажмете Delete, файлы будут удалены окончательно. Чтобы сразу удалить файлы без помещения в корзину, нажмите Shift+Delete.

Скрытые файлы. Все файлы и каталоги, чье название начинается с точки, в Linux считаются скрытыми. Как правило, Dolphin не показывает эти файлы. Чтобы они отображались, в меню с инструментами нужно выполнить команду Show Hidden Files (Отображать скрытые файлы). Еще быстрее включить отображение скрытых файлов можно сочетанием клавиш Alt+«+», и это же сочетание, нажатое снова, отключает их показ.

Права доступа. Чтобы любой пользователь не мог читать или изменять любые файлы и каталоги, какие ему вздумается, Linux сохраняет для каждого файла информацию о его владельце и правах доступа. Чтобы изменить данные о владельце или о правах доступа, щелкните на файле правой кнопкой мыши и выполните команду Properties ▸ Privileges (Свойства ▸ Права).

Доступ к носителям данных. На боковой панели Places (F9) содержится, в том числе, список сегментов диска, которые можно подключить к файловой системе щелчком кнопкой мыши. Если вы подключаете внешний дисковод по USB или Firewire, то на панели появляется соответствующее указание. Потом щелчком кнопкой мыши можно открыть файловый менеджер, в котором отобразится содержимое данного носителя. Прежде чем отключить устройство от компьютера, нужно либо выполнить в KDE команду контекстного меню Eject (Извлечь), либо открыть в боковой панели Dolphin перспективу Places (Места) с помощью F9 и выполнить команду контекстного меню Unmount (Извлечь). Только так вы можете гарантировать, что все файлы, которые оставались открытыми, правильно закроются, и в них не возникнет ошибок.

AudioCD. В программе Dolpin можно, в частности, просматривать список файлов, записанных на AudioCD. Для этого укажите адрес `ocd:/`. Особенность данной функции заключается в том, что все аудиотреки будут доступны также в форматах FLAC, MP3 (если у вас установлен пакет `lame`) и Ogg Vorbis. Теперь если скопировать эти файлы в каталог на жестком диске (методом перетаскивания), то аудиофайлы будут считаны и автоматически конвертированы в соответствующий формат. Если считывание диска длится слишком долго, можно деактивировать на вкладке Common (Общие) проверку ошибок. Это зачастую позволяет ускорить процесс в несколько раз, но заметно снижает качество аудио.

Доступ к сетевым каталогам. Доступ к локальной сети можно получить с помощью боковой панели Places (Места) или указав адрес `smb:/`. Для непосредственного доступа к конкретному каталогу на сервере Samba или Windows используйте вариант записи `smb://servername/sharename`. Такой способ записи следует применять и в тех случаях, когда Dolphin не может опознать в сети сервер Windows. Это часто происходит из-за настроек конфигурации сетевого экрана и самой сети.

После ввода имени пользователя и пароля Dolphin запрашивает информацию о том, с каким сервером требуется установить соединение — Windows или Samba. На некоторое время Dolphin запоминает это сочетание, но потом снова забывает — это делается по соображениям безопасности. Поэтому после значительного перерыва в работе приходится снова указывать имя пользователя и пароль. В модуле системных настроек Network & Connections ▸ Share (Сеть и соединения ▸ Совместное использование) можно задать стандартные значения имени пользователя и пароля.

ПРИМЕЧАНИЕ

Когда Dolphin не находит в локальной сети сервера Windows или Samba, в этом, возможно, виноват сетевой экран вашего дистрибутива. Например, в Fedora и SUSE стандартные настройки сетевого экрана не позволяют пользоваться сетевыми каталогами Windows. Чтобы справиться с этим неудобством, нужно правильно сконфигурировать сетевой экран.

FTP. Если адрес (URL) начинается с `ftp://`, то Dolphin автоматически переходит в FTP-режим. Активируемый при этом интерфейс и способы работы с ним

практически не отличаются от работы с файловым менеджером. Если вы хотите войти на FTP-сервер под определенным именем (а не анонимно), то используется способ записи `ftp://name@adress`. Как только будет установлено соединение с FTP-сервером, появится окно авторизации для ввода логина и пароля.

SSH. Dolphin также позволяет подключиться к другому компьютеру по безопасному протоколу SSH, обмениваться информацией с этим компьютером и копировать с него файлы. Для этого вводится адрес `fish://username@computername/`. После входа в систему Dolphin отобразит все файлы внешнего компьютера.

3.3. Konqueror и Rekonq

Программа Konqueror — это одновременно и файловый менеджер для опытных пользователей, и браузер, и сетевой клиент (протоколы FTP, SCP, каталоги Windows и т. д.), а также программа для просмотра документов (изображения, файлы справки и т. д.). Но из-за такого множества функций у Konqueror есть характерный недостаток — меню программы выглядит перегруженным, в нем очень сложно ориентироваться. Не считая этого неудобства, работать с Konqueror практически так же просто, как и с Dolphin. Далее по ходу главы я расскажу о различных дополнительных функциях этой программы. Подробная информация содержится на сайте http://www.konqueror.org/.

Использование программы в качестве файлового менеджера

Чтобы использовать Konqueror как файловый менеджер, задайте в адресной строке желаемый каталог. В меню View (Вид) можно выбрать один из тех же вариантов представления, которые действуют и в Dolphin. Konqueror может создавать эскизы (картинки для предварительного просмотра) для всех файлов, формат которых он распознает. Это делается с помощью команды View ▸ Preview (Вид ▸ Предварительный просмотр). Размер таких эскизов при соответствующем режиме просмотра можно изменять сочетаниями клавиш Ctrl+ или Ctrl-.

С помощью клавиши F9 можно выводить на экран и вновь отключать боковую панель навигации (sidebar). На ней можно отображать закладки, устройства, дерево каталогов, сетевой обозреватель и другие вспомогательные навигационные средства.

Очень красиво организован доступ к архивам. Если вы щелкаете на архиве в формате ZIP или TAR (то есть на файле `name.tar`, `name.tgz` или `name.zip`), содержимое этого архива отображается в новом каталоге, прямо в программе Konqueror.

Работа с жесткими дисками. Если вы хотите узнать, в каком из ваших каталогов находятся самые объемные файлы, используйте режим Konqueror File Sizes (Размеры файлов). В этом режиме Konqueror создает рисунок из вложенных друг в друга прямоугольников, размер каждого из которых соответствует размеру файлов. Если в вашей системе такой режим не предоставляется, то нужно установить соответствующий дополнительный пакет (плагин) для Konqueror. (В Ubuntu соответствующий пакет называется `konq-plugins`.)

Цветовое оформление, глубина рекурсии и другие детали оформления можно настроить в меню View (Вид). Щелчком кнопкой мыши можно перейти в подкаталог, чтобы подробнее ознакомиться с его содержанием. Кроме того, чтобы попасть в окно для просмотра размера файлов, можно выполнить команду `fsview`. В таком случае рисунок отобразится без меню Konqueror и других параметров.

Протоколы KIO (ввод-вывод в Konqueror). В адресной строке Konqueror и Dolphin можно указывать веб-адреса, имена файлов и т. д. Konqueror, в принципе, отображает все данные в браузере, а Dolphin при работе с некоторыми протоколами открывает внешнюю программу. Чтобы Konqueror и Dolphin знали, как интерпретировать адрес, этот адрес должен идти после обозначения протокола (табл. 3.1). Обратите внимание, что иногда при записи используются два слэша, иногда — один, а в некоторых случаях — ни одного. За обработку этих протоколов отвечают модули, которые называются в KDE-номенклатуре KIO Slaves (где slave — ведомый, KIO — ввод-вывод KDE).

Таблица 3.1. Важнейшие протоколы ввода-вывода в KDE

Протокол	Значение
file:/etc/fstab	Локальный файл
tar:/archivdatei	Доступ к архивному файлу TAR
audiocd:/	Доступ к AudioCD
trash:/	Удаленные файлы (корзина)
http://www.kofler.info	Веб-сайт
ftp://user@mars/folder	FTP-сервер на компьютере mars
sftp://user@mars/folder	SFTP-сервер на компьютере mars
fish://user@mars/folder	SSH-доступ к компьютеру mars
smb://mars/myshare	Сетевой каталог Windows
man:ls	Справочная страница по команде ls
info:emacs	Информация по программе emacs
help:kmail	Справка KDE по программе kmail
applications:/	Список всех программ
fonts:/	Список всех шрифтов
remote:/	Общий обозреватель сети
settings:/	Конфигурационные модули центра управления

Использование в качестве веб-браузера

Сравнение KHTML и Webkit. Отображение сайтов в Konqueror может осуществляться в зависимости от конфигурации, либо собственным механизмом визуализации Konqueror, называемым KHTML, либо с помощью WebKit. Разработчикам KDE все хуже удается обеспечивать совместимость KHTML с постоянно обновляющимися стандартами HTML и Веба в целом. Чтобы настроить механизм визуализации, выполните в консоли следующую команду как обычный пользователь (но не как администратор `root`!):

```
user$ keditfiletype text/html
```

В диалоговом окне Edit File Type (Изменить тип файла) перейдите на вкладку Embed (Встроить), далее выберите служебную программу Webkit, переместите ее в самую высокую позицию в иерархии и закройте окно, нажав кнопку OK. После перезапуска системы Konqueror уже будет обращаться к WebKit. Таким образом, это будет соответствовать системе стандартизации ACID3 (http://ru.wikipedia.org/wiki/Acid3 — *Примеч. перев.*). В openSUSE такая настройка действует по умолчанию. Kubuntu же по-прежнему работает с KHTML. Переход на работу с Webkit возможен лишь при условии, что в вашей системе установлен пакет `kpart-webkit`.

Настройки. Пожалуй, больше не найдется браузера, в котором можно было бы изменять такое множество настроек. Выполнив команду Settings ▸ Konqueror Setup (Настройки ▸ Настройка Konqueror), вы попадаете в диалоговое окно с многочисленными модулями. При этом некоторые из модулей сами состоят из нескольких диалоговых окон!

Наряду с поиском в Интернете по текстовому запросу, вводимому в специальное поле — такой механизм действует во многих браузерах, — поиск можно выполнять и прямо в адресной строке. Для этого определяются специальные сокращения. Например, при вводе сокращения `gg:abc` запускается поиск по запросу `abc` на сайте http://www.google.com. В диалоговом окне Web Shortcuts (Веб-аббревиатуры) можно дополнить имеющийся список новой аббревиатурой.

Плагины (подключаемые модули). Как только вы установите программу `nspluginviewer`, которая обычно находится в пакете `konqueror-nsplugins`, Konqueror сможет работать с теми же плагинами, что и Firefox. Если какие-то плагины не работают, откройте диалоговое окно Extensions (Расширения). Там можно посмотреть, какие плагины обнаружил Konqueror, и указать, в каких каталогах следует искать новые плагины. С помощью кнопки Search for new plugins (Искать плагины) можно добавить недавно установленные плагины в список плагинов Konqueror.

Кроме интерфейса, предназначенного для взаимодействия с плагинами Firefox, Konqueror поддерживает и собственные плагины, специфичные для KDE. К наиболее популярным функциям системы относится автоматический перевод вебсайтов, функция автоматического обновления, позволяющая перезагружать сайт после истечения определенного временного промежутка, а также просмотрщик объектной модели документа, визуализирующий структуру HTML-страницы. Зачастую эти плагины требуется устанавливать отдельно. В большинстве дистрибутивов плагины объединены в пакете `konq-plugins`.

Java. KDE не работает с плагином Java как таковым, а использует KJAS (сервер аплетов Java для KDE, файл `kjavaappletviewer.so`). KJAS, в свою очередь, запускает интерпретатор Java — специальную программу под названием `java`. Если Konqueror не удается найти этот файл, нужно узнать полное имя файла с помощью команды `which java` и внести данное имя в конфигурационное диалоговое окно Konqueror, которое называется JAVA AND JAVASCRIPT (Java и JavaScript).

Rekonq

Если меню Konqueror кажется вам слишком перегруженным, рекомендуется обратить внимание на браузер Rekonq, который хорошо подходит для работы с KDE. По умолчанию этот браузер базируется на WebKit. Если вы параллельно работае-

те и с Konqueror, то вас, несомненно, обрадует тот факт, что оба браузера используют одну и ту же систему для хранения закладок. Если закладка была определена в Konqueror, то ею можно будет пользоваться и в Rekonq (и наоборот). Новые закладки создаются сочетанием клавиш Ctrl+B. Вертикальная панель закладок включается или выключается сочетанием клавиш Shift+Ctrl+B.

Еще одна общая черта Rekonq и Konqueror — поддержка веб-аббревиатур. Если, например, ввести команду `wp:MBR`, то будет запущен поиск в «Википедии» о главной загрузочной записи (MBR). Rekonq оснащен встроенным фильтром для отключения рекламы, который настраивается командой Rekonq settings ▸ Ad Blocker (Настройки Rekonq ▸ Фильтр рекламы). При необходимости этот фильтр также можно отключить.

3.4. Конфигурация

Различные конфигурационные модули KDE представлены в области управления системой (команда `systemsettings`) (рис. 3.5). Поскольку найти нужный модуль иногда бывает непросто, разработчики KDE снабдили эту программу поисковой функцией, обеспечивающей поиск по ключевым словам (например, `Window`). Из модуля, активного в настоящий момент, вы попадаете в окно обзора модулей с помощью кнопки Overview (Обзор).

Рис. 3.5. Центр управления KDE

Отдельные модули центра управления можно также вызывать в форме `kcmshell4 имямодуля`. Чтобы увидеть список предоставляемых на выбор модулей, выполните команду `kcmshell4 --list`. При работе с модулями учитывайте, что измененные настройки вступают в силу только после того, как изменения будут подтверждены кнопкой Apply (Применить).

Как и в Gnome, в KDE содержатся конфигурационные модули, касающиеся не Рабочего стола как такового, а системных настроек (работа с сетью, с принтером и т. д.). Эти модули очень полезны, в первую очередь, в тех дистрибутивах, которые сами не содержат специальных конфигурационных инструментов. При наличии всегда следует работать именно с теми конфигурационными программами, которые предназначены для конкретного дистрибутива. Время от времени модули вступают в конфликт с дистрибутивом — либо из-за особенностей самого дистрибутива, либо потому, что эти модули просто устарели. Возможно, перед началом работы потребуется войти в систему с административными правами и сначала перевести в административный режим работы те модули, которые отвечают за системные настройки.

Конфигурационные каталоги. Большинство программ KDE хранит свои настройки в файлах каталога `~/.kde/` или `~/.kde4/`. Там, в частности, существуют следующие подкаталоги:

```
~/.kde[4]/Autostart/     (Персональные программы автозапуска)
~/.kde[4]/share/config/  (Настройки)
~/.kde[4]/share/apps/    (Другие файлы, специфичные для конкретной программы)
```

Некоторые программы KDE соответствуют стандарту XDG и применяют каталоги XDG, перечисленные в предыдущей главе:

```
~/.cache/                (Кэш)
~/.config/progname/      (Настройки)
~/.local/share/progname/ (Пользовательские данные)
```

Трехмерные графические эффекты в настольной системе. В KDE за трехмерные эффекты Рабочего стола отвечает оконный менеджер KWin. Для настройки таких эффектов применяется модуль системных настроек Workspace Effects (Эффекты рабочей области). При первоначальной установке активны лишь немногие эффекты, но их существует великое множество. Трехмерные эффекты функционируют лишь при условии, что в системе установлены требуемые графические драйверы.

Автоматический вход в систему. После запуска компьютера вам нужно войти в систему, прежде чем вы сможете приступить к работе. Если вы — единственный пользователь компьютера и можно не опасаться, что другие лица получат к нему доступ, то первый вход в систему, происходящий сразу после запуска компьютера, можно автоматизировать. Функцией автоматического входа можно управлять в модуле системных настроек Login Screen (Экран для системных настроек). Там следует установить флажок Auto-Login (Автоматический вход в систему) и выбрать пользователя, вход которого в систему должен осуществляться автоматически. «За кулисами» системы за вход обычно отвечает дисплейный менеджер KDE. Например, в Kubuntu для этого применяется программа LightDM.

Если на вашем компьютере установлены как Gnome, так и KDE, то за вход в систему, возможно, будет отвечать дисплейный менеджер Gnome. Чтобы узнать, какой именно дисплейный менеджер работает в системе, проще всего выполнить в окне консоли следующую команду:

```
user$ ps ax | egrep 'gdm|kdm|lightdm'
```

В дистрибутивах Novell и SUSE настройка функции автоматического входа в систему протекает независимо от самого Рабочего стола и осуществляется в файле `/etc/sysconfig/displaymanager`. Не пробуйте изменять функцию автоматического входа с помощью Gnome или KDE: при первом же удобном случае ваши настройки будут вновь переписаны YaST!

Автозапуск. При каждом выходе из системы завершаются все работавшие программы. При следующем входе в систему KDE не просто перезапускает программы, которые работали на момент выхода из системы, но и восстанавливает последнюю сессию. В случае с программами KDE этот механизм обычно работает хорошо, но если речь идет о других программах, то либо возникают определенные ограничения (последние использовавшиеся документы не загружаются), либо никакого восстановления вообще не происходит.

Детали этого функционала задаются в модуле системных настроек Start and Quit ▸ Session Management (Запуск и завершение ▸ Управление сессиями). Последнее актуальное состояние сессии сохраняется в файлах каталога `~/.kde[4]/ share/config/session`.

Независимо от применяемой системы управления сессиями, в каталоге `~/.kde[4]/Autostart` можно указать программы, которые будут запускаться после каждого входа в систему. KDE ожидает найти в этом каталоге файлы `*.desktop`, описывающие программу, которую необходимо запустить. Проще всего создавать такие файлы, открыв каталог `~/.kde/Autostart` в файловом менеджере Dolphin и скопировав туда методом перетаскивания желаемую программу прямо из меню KDE. В качестве альтернативы, конфигурацию также можно осуществлять в системном модуле Start and Quit ▸ Autostart (Запуск и завершение ▸ Автозапуск).

Если вы параллельно применяете оба механизма, то есть управление сессиями и каталоги автозапуска, то не исключено, что недавно использовавшаяся программа будет запущена дважды. Обратите внимание: в KDE предусмотрено несколько каталогов для автозапуска:

```
~/.kde[4]/Autostart/  (Персональные программы для автозапуска)
/usr/share/autostart/ (Глобальные программы для автозапуска в KDE)
/etc/xdg/autostart/   (Глобальные программы для автозапуска в Gnome и KDE)
```

Разрешение экрана и конфигурация с двумя мониторами. В модуле Display and Monitor (Отображение и монитор) вы указываете, должны ли использоваться несколько мониторов и сигнальных выходов (если да — то сколько) и с каким разрешением экрана вы хотите работать. Для сохранения новых настроек нужно нажать кнопку Save as Default (Сохранить по умолчанию).

Внешний вид Рабочего стола. Существует масса возможностей настройки внешнего вида (оптического представления) Рабочего стола. Если у вас есть время и желание, можно посвятить настройке Рабочего стола многие часы.

- **Фон Рабочего стола.** Для настройки фонового изображения щелкните на Рабочем столе правой кнопкой мыши и выполните команду Desktop Settings (Настройки Рабочего стола). После этого вы сможете настроить цвет Рабочего стола и фоновое изображение для него.
- **Дизайн Рабочего стола.** В модуле Workspace Appearance (Внешний вид Рабочего стола) есть вкладка Workspace Design ▸ Workspace (Дизайн рабочей области ▸ Рабочая область), с помощью которой можно настроить дизайн (тему) для Рабочего стола. Дизайн определяет базовые настройки внешнего вида панелей, меню KDE, декорирование окон и т. д., а также применяемую при этом цветовую гамму. С помощью команды Download New Design (Загрузить новый дизайн) можно скачать дополнительные варианты дизайна рабочих областей для KDE (они находятся на сайте http://kde-look.org/), а потом активизировать. Не забудьте, что новый дизайн рабочей области будет применен только после нажатия кнопки Apply (Применить). Мне, например, очень нравится дизайн Caledonia.
- **Оформление элементов управления.** В модуле системных настроек Application Appearance (Отображение приложений) есть вкладка Stil (Стиль), позволяющая выбирать из нескольких вариантов оптического представления и компоновки кнопок, полей с настройками, полос прокрутки изображений и др. Но большинство предлагаемых вариантов — довольно старые и выглядят они тоже старомодно.
- **Оформление окон.** В модуле системных настроек Workspace Appearance (Отображение рабочего пространства) есть диалоговая вкладка Window Decoration (Декорирование окна), где можно выбрать один из нескольких макетов окна. При этом также меняется цвет и оформление рамки окна. Стандартные макеты не блещут оригинальностью, но вы без труда можете загрузить и опробовать другие варианты декорирования.

  Кнопки Setup Window Decoration (Настройка декорирования окна) и Setup Buttons (Настройка кнопок) позволяют изменять и некоторые другие детали применяемого варианта декорирования, в частности, обрамление и тени окна, расположение оконных кнопок и т. д.
- **Цвета.** Цветовое оформление рамки окна, меню, панелей и т. д. задаются на уровне дизайна Рабочего стола и варианта декорирования окон. Модуль системных настроек Application Appearance (Отображение приложений) позволяет настраивать на вкладке Colors (Цвета) собственные цветовые схемы. К сожалению, существующие в системе диалоговые вкладки довольно неудобны в работе. Зачастую не удается узнать, как отразится на системе то или иное изменение, если не испробовать его на практике. Гораздо удобнее воспользоваться командой Download New Color Schemes (Загрузить новые цветовые схемы) и взять все цветовые настройки прямо из Интернета в готовом виде.

Конфигурация принтера. В модуле центра управления KDE Printer Setup (Настройка принтера) осуществляется конфигурация принтеров (рис. 3.6). В openSUSE для настройки принтера лучше применять YaST. Поэтому модуль KDE, предназначенный для конфигурации принтера, не отображается в диалоговом окне

настройки системы, но его можно запустить с помощью команды `kcmshell printers`.

Рис. 3.6. Конфигурация принтера в KDE

Для запуска ассистента, настраивающего новый принтер, выполните команду New Printer (Новый принтер). На первом этапе указывается тип принтера (например, Network Printer (Сетевой принтер)), на втором — производитель принтера и модель устройства. Чтобы завершить конфигурацию, необходимо присвоить принтеру имя. В поле Address (Адрес) вы также можете указать физическое расположение принтера. Это удобно, если в сети действует несколько принтеров, установленных в разных офисах или на разных этажах.

После этого принтер может использоваться всеми программами Linux. Если в данный момент принтер недоступен и уже накопилось несколько задач, связанных с распечаткой, они обслуживаются в форме очереди.

Пиктограмма для входа. В зависимости от конфигурации экрана для входа в систему каждому пользователю соответствует на нем маленькая картинка. Эту картинку можно настроить или изменить в системном модуле User Account Details (Пользовательские учетные записи). В диалоговом окне можно изменить пароль, а также многие другие настройки. Пиктограмма для входа сохраняется в под именем `~/.face.icon`, в формате PNG.

MIME. В Konqueror при двойном щелчке кнопкой мыши на файле в формате OGG открывается программа Amarok. Это происходит благодаря настройкам MIME, действующим в KDE. Аббревиатура MIME расшифровывается как «многоцелевые расширения интернет-почты». Эта служба представляет собой своего рода базу данных, обеспечивающую соотнесение файлов определенных форматов с теми или иными программами. Список типов файлов MIME можно просмотреть и изменить в модуле системных настроек File Associations (Соотнесение файлов). Некоторым типам файлов может соответствовать несколько программ. Если файл открывается щелчком левой кнопкой мыши, то используется программа, стоящая первой

в иерархии. Все остальные приложения предлагаются на выбор, если вы щелкаете на файле правой кнопкой мыши и выбираете команду Open with (Открыть с помощью).

Тени. По умолчанию активное окно выделяется прозрачной голубоватой подсветкой. Окна, которые в данный момент неактивны, также имеют ярко выраженную тень. Разумеется, в KDE можно настраивать и тени, но соответствующее диалоговое окно найти не так просто. Чтобы попасть в него, нужно перейти в окне Workspace Appearance (Отображение рабочего пространства) на вкладку Window Decoration (Декорирование окна). Кнопка Setup Window Decoration (Настроить декорирование окна) открывает следующее окно, в котором есть вкладка Shadows (Тени), где и расположены интересующие нас настройки.

Настройка программ по умолчанию. По умолчанию в KDE в качестве веб-браузера используется Konqueror, KMail или Kontact — в качестве почтового клиента, а `konsole` — для работы с консолью. Если вы хотите, чтобы при нажатии соответствующих ссылок в KDE запускались другие программы, то необходимые настройки можно внести в модуле Default Components (Компоненты по умолчанию).

Оконный менеджер. Для управления окнами, то есть для перемещения, увеличения, уменьшения окна, применяется оконный менеджер. В KDE для этого по умолчанию применяется программа KWin. Многочисленные настройки можно изменять в модуле системных настроек Window Behavior (Поведение окна). Здесь, в частности, можно указать, как будут действовать кнопки мыши в титульной строке окна, в соответствии с какими критериями будут размещаться на экране новые окна, будут ли окна фиксироваться при перемещении и т. д.

Многие операции с онами можно осуществлять и с клавиатуры. Предусмотренные для этого сочетания клавиш находятся в модуле системных настроек Shortcuts and Gestures (Сочетания клавиш и жесты), диалоговая вкладка Default Shortcuts (Сочетания клавиш по умолчанию). Здесь можно просматривать и менять соответствующие настройки.

3.5. Запись CD/DVD с помощью K3b

K3b — это универсальная программа для записи дисков, присутствующая в Linux в настоящий момент. Функциональное разнообразие программы не оставит равнодушными даже закоренелых сторонников Gnome. Немного портят картину лишь меню и диалоговые окна настроек, в которых иногда довольно сложно ориентироваться. Но не пугайтесь! Для решения стандартных задач — например, для записи CD с резервными копиями файлов — многие параметры не важны, и их спокойно можно игнорировать. Как правило, в K3b по умолчанию действуют самые адекватные настройки.

Запись CD и DVD с данными. В зависимости от того, хотите ли вы записать CD или DVD, выберите вид диска после выполнения команд File ▸ New Project ▸ New Data Project (Файл ▸ Новый проект ▸ Новый проект с данными). После этого нужно переместить каталоги и файлы, для которых создаются резервные копии, из дерева каталогов (сверху) в область CD/DVD (внизу). В нижней части окна ото-

бражается индикатор протекания процесса, демонстрирующий, сколько места требуется на запись выбранных файлов.

Кнопка Burn (Запись), расположенная в нижней части окна K3b, открывает диалоговое окно, в котором можно сделать различные настройки. Как правило, параметры можно оставить заданными по умолчанию и, наконец, начать процесс записи, снова нажав кнопку Burn (Запись). Во время записи в диалоговом окне отображается текущее состояние.

Возможности записи. Рекомендуется перепроверять после записи, корректно ли записались данные на CD или DVD (для этого используется команда Check Recorded Data (Проверить записанные данные)). Проверка занимает примерно столько же времени, сколько и сама запись. Ничего, безопасность лишней не бывает! Именно при записи DVD проблемы возникают достаточно регулярно. (В случае появления таких проблем, можно снизить скорость записи DVD в два раза от максимальной.)

Если вы не уверены, обеспечивает ли ваша система достаточную скорость записи CD или DVD, можно сначала протестировать процесс записи (флажок Simulate (Имитировать)). Хотя при этом и не записывается никаких данных, в приводе должен находиться пустой CD или DVD. С самим диском в этом случае ничего не происходит.

Обычно данные считываются непосредственно с жесткого диска и сразу же записываются на CD или DVD. Если при этом возникнут ошибки, установите флажок Create Image (Создать образ). В таком случае по результатам записи K3b создает единственный файл, в котором содержатся все необходимые данные. Затем данный файл записывается повторно. Поскольку единственный файл считывается быстрее, чем множество мелких файлов, такой метод более надежен.

Если вы хотите дать имя CD или DVD, то это можно сделать на вкладке File System (Файловая система) в поле Medium Name (Имя носителя).

K3b записывает все файлы на диск таким образом, что впоследствии любой пользователь Linux может читать все файлы. Если вы хотите сохранить исходные права доступа, то на вкладке File System (Файловая система) нужно нажать кнопку Custom (Пользовательские настройки), а затем установить флажок Preserve File Permissions (Сохранить права доступа к файлам).

Обычные CD и DVD можно записывать только один раз. Если вы хотите записать относительно небольшое количество данных, то, конечно же, неразумно всякий раз тратить на это новый CD или DVD. Для таких случаев существует метод записи в несколько сеансов (Multi-Session). Его основная идея заключается в том, что имеющиеся данные не изменяются и на диск могут записываться новые данные, если на диске еще есть свободное место.

K3b позволяет записывать в несколько сессий как CD, так и DVD. В любом случае нужно задать правильные настройки на вкладке Settings (Настройки).

- Первая сессия: если вы начинаете запись CD/DVD в режиме нескольких сессий, то на вкладке Miscellaneous (Разное) можно выбрать настройку Start Multi-Session (Начать запись в режиме нескольких сессий).
- Следующие сессии: при всех следующих сессиях выбирается настройка Continue Multi-Session (Продолжить запись в несколько сессий).

- Последняя сессия: чтобы окончательно завершить запись («закрыть диск»), выберите настройку Finalize Multi-Session (Завершить мультисессионную запись). После этого запись данных на диск становится невозможной.

Если вы сохраните один и тот же файл в нескольких сессиях, то он несколько раз будет записан на CD или DVD. При чтении диска автоматически просматривается самая новая версия файла (то есть записанная в ходе наиболее новой сессии).

Копирование CD/DVD. Чтобы копировать CD или DVD (при этом неважно, записаны на CD обычные данные или аудиоинформация), выполните команду Extras ▸ Copy Medium (Дополнительно ▸ Скопировать носитель). Как правило, достаточно просто щелкнуть на кнопке Start (Пуск) — K3b сам применит все необходимые настройки. Изменять параметры приходится только в отдельных случаях. Обратите внимание, что у вас должно быть достаточно места в каталоге для временных файлов, чтобы весь носитель можно было записать в буфер обмена.

Если у вас в системе два привода, то укажите в качестве Source Medium (Исходный носитель) второй дисковод. Если у вас оптимально подобрано аппаратное обеспечение, то можно также снять флажок Create Image (Создать образ). В таком случае данные с исходного диска будут считываться и сразу записываться на второй диск. При этом необходимо учитывать, что подробное считывание данных непосредственно с AudioCD длится достаточно долго, а результаты такого процесса не всегда удовлетворительны. Обычно все-таки бывает целесообразно создавать образ диска (команда Create Image (Создать образ)) и рассчитывать, что придется довольно долго подождать.

При работе с CD K3b различает два режима копирования: Normal Copy (Нормальная копия) и Clon Copy (Клонированная копия). Внутри системы для выполнения первого и второго процессов применяются разные программы. Как правило, при создании нормальной копии получаются удовлетворительные результаты. Только в особых случаях — например, при копировании VideoDVD — нужно делать и клонированную копию.

Запись ISO-образа диска. Для записи ISO-образа выполните команду Extras ▸ Burn Image (Дополнительно ▸ Запись образа). В диалоговом окне для записи выберите файл, который необходимо записать. Обычно этот файл имеет расширение ISO. После этого K3b автоматически рассчитывает для файла контрольную сумму MD5. По умолчанию контрольная сумма предоставляется и на сайтах, на которых можно скачивать ISO-образы. Таким образом, вы можете удостовериться, что полученный вами файл в точности соответствует оригиналу.

3.6. Программы KDE

В этом разделе представлены наиболее часто используемые программы для KDE.

Файловый архив (Ark). Программа Konqueror отлично подходит для просмотра архивов в форматах TAR.GZ, TGZ и ZIP, а также для извлечения файлов из них. Но в Konqueror вы не можете создавать новые архивы, а также изменять уже имеющиеся. Для решения именно таких задач предназначена программа Ark, функционально напоминающая, например, архиватор WinZip.

Удаленное обслуживание. Если на вашем компьютере возникает проблема — например, какая-то программа функционирует неправильно — то лучше всего обратиться к другу, который разбирается в Linux и может оказать вам помощь с удаленной машины. Для этого запускается VNC-клиент для удаленного администрирования (VNC — Virtual Network Computing). Для этого выполните команду Programs ▸ System ▸ Share Workspace (Программы ▸ Система ▸ Совместное использование рабочего пространства). В Kubuntu для этого сначала требуется установить программу `krfb`, обслуживающую такие соединения.

Ваш помощник может воспользоваться любым VNC-клиентом или специальной программой `krdc` для KDE. Теперь в специальном окне на своем компьютере он будет видеть все содержимое вашего экрана. Пользуясь мышью и клавиатурой своего компьютера, он сможет работать со всеми вашими программами.

Работа с консолью. С помощью программы `Konsole` можно запускать одну или несколько оболочек (командных окон) и представлять их вывод в текстовых консолях. Для перехода между консолями используются сочетания клавиш Shift+←, либо Shift+→. С помощью Ctrl+Shift+← или Ctrl+Shift+→ можно менять последовательность вкладок, а двойным щелчком кнопкой мыши можно переименовывать вкладки. Кроме того, в Konqueror очень удобно перемещать имена файлов в консоль методом перетаскивания. Так вы избавляетесь от массы печатной работы.

KRunner. Сочетание клавиш Alt+F2 запускает крошечную программку KRunner. Она, в свою очередь, позволяет запустить любую другую программу — ту, чье имя вы в нее введете. Но KRunner способна на большее: с ее помощью можно, например, производить вычисления (вводим `2x3`, получаем `6`), отображать файлы справки `man` (вводим `man:ls`), использовать веб-аббревиатуры Konqueror (например, для поиска в Google вводим `gg:kde`), переходить в другую сессию (команда `switch username`), активизировать другой профиль энергопотребления (`power profilname`) и т. д.

Управление паролями. Если вы впервые вносите учетные данные в Konqueror (например, для работы с почтовым сервисом, таким как mail.ru), появляется ассистент, помогающий настроить так называемый цифровой бумажник (digital wallet). Программа KWallet обеспечивает безопасное управление данными формы. Если позже вы откроете в Интернете эту же страницу, KWallet заполнит данные формы за вас. KWallet также используется и другими программами, в частности, KMail/Kontact, NetworkManager и многими другими. Доступ к этим данным, которые хранятся в зашифрованном виде, защищаются мастер-паролем. Этот пароль в ходе каждой сессии KDE нужно вводить только один раз.

PDF и PostScript. При щелчке кнопкой мыши на файле PDF или PS открывается программа Okular, отображающая документ. Теперь можно просматривать документ, распечатывать отдельные страницы и т. д.

Управление шифрованием. Если вы скрепляете ваше электронное письмо цифровой подписью, зашифровываете его или используете другую программу, работающую с центральным PGP-ключом, вам очень пригодится инструмент для централизованного управления ключами. В KDE эта задача решается с помощью программы KGpg. За управление сертификатами S/MIME отвечает программа Kleopatra (раньше она называлась KGpgCertmanager). Во многих дистрибутивах программа Kleopatra содержится в пакете `kdepim`.

Текстовые редакторы (KEdit, KWrite, Kate). В некоторых дистрибутивах вместе с KDE устанавливаются сразу три текстовых редактора: KEdit, KWrite и Kate. KEdit и KWrite очень удобны в использовании, но их функционал ограничивается только самыми простыми возможностями. Kate рассчитан на программиста или на опытного пользователя. В нем можно объединять несколько открытых файлов в одну «сессию», объединять фрагменты кода (например, функции или классы) и т. д.

Буфер обмена (Klipper). Как правило, в KDE задействуется программа Klipper, которая по умолчанию запоминает семь последних информационных фрагментов, записанных в буфер обмена. Щелкнув кнопкой мыши в системном разделе, одну из этих записей можно снова сделать активным содержимым буфера обмена, а более старые записи могут при этом использоваться как обычно. При необходимости можно активировать Actions (Действия). После этого Klipper анализирует все новые записи и автоматически выполняет соответствующие им операции. Например, всякий раз, когда был выделен HTTP-адрес, автоматически открывается окно Konqueror с соответствующей веб-страницей. Но лично мне работа с Klipper никогда не давалась, поэтому я отключил автозапуск этой программы (в меню есть для этого специальная команда Quit (Завершить)).

Переименование файлов (Krename). Программа Krename позволяет удобно переименовывать файлы, присваивать им порядковые номера, добавлять к имени файла дату и т. д. При переименовании даже могут применяться регулярные выражения. Прежде чем файл будет действительно изменен, программа позволяет предварительно просмотреть старое и новое имя. Krename — просто незаменимый инструмент, если требуется заново организовать большое собрание аудиофайлов или изображений! Во многих дистрибутивах эту программу требуется установить (соответствующий пакет называется `krename`).

4 VirtualBox

Виртуализация позволяет параллельно использовать на одном компьютере несколько операционных систем. Эта возможность очень востребована на практике: можно установить Linux в Windows, выполнять Windows в Linux, тестировать новую альфа-версию дистрибутива xyz, не опасаясь повредить действующую (стабильную) версию Linux, уверенно отделять друг от друга функции сервера (виртуализация сервера) и т. д.

Эта глава начинается с обзора основных функций и программ, связанных с виртуализацией. Далее основное внимание уделено программе VirtualBox. Для серверной виртуализации лучше подходит программа KVM.

4.1. Основы виртуализации

В этом разделе объясняется, почему существует так много программ для виртуализации, на основе каких технологий они работают и какая программа лучше подходит для выполнения определенных целей. Если ваша приоритетная задача — быстро настроить виртуальную машину Windows для работы в Linux либо выполнить Linux в системе Windows, то можете пропустить этот вводный раздел и перейти к разделу о VirtualBox.

Технологии виртуализации

Гость и хозяин. При описании систем виртуализации закрепилась метафора, рассматривающая основную систему как хозяина (host), а работающие на ней виртуальные машины как гостей (guests).

Технологии. Существуют различные методы виртуализации операционных систем. В следующем списке перечислены наиболее распространенные из них и названы некоторые программы (фирмы), которые пользуются этими технологиями.

- **Полная виртуализация (виртуальные машины, эмуляция).** В данном случае программа имитирует работу виртуального аппаратного обеспечения, то есть компьютера, состоящего из процессора, ОЗУ, жесткого диска, сетевой карты и т. д. Гостевые системы «считают», что виртуальное аппаратное обеспечение

является реальным. Чтобы такая система функционировала, работающая на хозяине программа виртуализации должна отслеживать код гостя и заменять определенные команды другими фрагментами кода. Эту задачу выполняет гипервизор (Virtual Machines Monitor, VMM — «монитор виртуальных машин»). Такая программа-гипервизор также отвечает за события, связанные с хранением информации и управлением процессами.

Преимущества: на виртуальной машине может функционировать практически любая операционная система. При этом в операционную систему не требуется вносить никаких изменений.

Недостатки: работает сравнительно медленно.

Программы/фирмы: VMware, QEMU, Parallels, VirtualBox, Microsoft Virtual PC.

- **Паравиртуализация.** В данном случае хозяин также же предоставляет виртуальные машины, на которых выполняются программы гостей. Отличие от полной виртуализации состоит в том, что гостевую операционную систему для виртуализации требуется модифицировать, после чего эта система напрямую сообщается с VMM.

  Преимущества: высокая эффективность.

  Недостатки: требует специальной модификации операционных систем для целей виртуализации. Для такой системы с открытым кодом, как Linux, это не составляет никакой проблемы, чего не скажешь о коммерческих операционных системах, например Windows. (В этой области налажена совместная работа между Xen и Microsoft или Novel и Microsoft, поэтому вероятно, что в будущем появятся версии Windows Server, оптимизированные специально под Xen.)

  Программы/фирмы: Xen, UML (Linux в пользовательском режиме).

- **(Пара)виртуализация с поддержкой аппаратного обеспечения.** Современные процессоры производства Intel и AMD содержат аппаратные функции, предназначенные для упрощения процессов виртуализации. В Intel такая технология называется Intel-VT (ранее — Vanderpool), а в AMD — AMD-V (ранее — Pacifica).

  Преимущества: высокая эффективность, при некоторых вариантах внедрения не требуется вносить изменения в операционную систему.

  Недостатки: необходимы специальные процессоры.

  Программы/фирмы: KVM, Xen.

- **Виртуализация на уровне операционной системы (контейнеры).** При использовании данного метода настоящие виртуальные машины не применяются. Вместо этого при таком подходе машины применяют общее ядро и фрагменты файловой системы хозяина. К важнейшим задачам системы виртуализации относится, в частности, обеспечение изоляции между хозяином и гостями для исключения каких бы то ни было проблем с безопасностью.

  Достоинства: очень эффективна, сберегает ресурсы (ОЗУ, дисковое пространство и т. д.).

Недостатки: может применяться только тогда, когда хозяин и гости используют в точности одну и ту же операционную систему и совершенно одинаковую версию ядра. Операционная система должна быть модифицирована соответствующим образом.

Программы/фирмы: OpenVZ, Virtuozzo, Linux-VServer.

Все перечисленные методы, кроме первого, требуют внесения изменений в ядро, причем соответствующие операции производятся через Linux. В настоящее время, по крайней мере официально, в состав ядра входят только те функции виртуализации, которые относятся к KVM и UML. При использовании других методов ядро необходимо модифицировать с помощью неофициальной заплатки. Если, например, вы работаете с Xen-образным дистрибутивом, то знайте, что дистрибьютор заблаговременно встраивает в ядро функции, необходимые для работы в Xen. Более подробную информацию о различных технологиях виртуализации можно прочитать в «Википедии», а также на следующих сайтах: http://virt.kernelnewbies.org/TechOverview; http://wiki.openvz.org/Introduction_to_virtualization.

Поддержка процессора. Чтобы убедиться, что ваш процессор (Intel-Vt или AMD-V) поддерживает аппаратную виртуализацию, выполните команду `egrep`. Приведенный здесь результат получен на процессоре Intel-i7. Если в результате открывается пустое окно, это означает, что процессор не поддерживает виртуализацию либо эта функция была деактивирована в BIOS/EFI.

```
user$ egrep '^flags.*(vmx|svm)' /proc/cpuinfo
flags : fpu vme de pse tsc msr pae mce cx8 apic mtrr pge mca cmov pat pse36
      clflush dts acpi mmx fxsr sse sse2 ss ht tm pbe syscall nx rdtscp lm
      constant_tsc arch_perfmon pebs bts rep_good xtopology nonstop_tsc
      aperfmperf pni pclmulqdq dtes64 monitor ds_cpl vmx est tm2 ssse3 cx16
      xtpr pdcm sse4_1 sse4_2 popcnt aes xsave avx lahf_lm ida arat epb xsaveopt
      pln pts dts tpr_shadow vnmi flexpriority ept vpid
...
```

Виртуальное аппаратное обеспечение

Эмулирование виртуального аппаратного обеспечения — это очень сложный процесс. В зависимости от механизма виртуализации или варианта его внедрения вы рано или поздно столкнетесь с границами возможностей вашего компьютера.

ОЗУ. Память компьютера должна быть достаточно объемной, чтобы выполнять все требования ресурсов хозяина и гостей, работающих на нем. Чем больше систем должны функционировать одновременно, тем больше оперативной памяти требуется. Например, на компьютере, которым я пользуюсь для тестирования, объем оперативной памяти достигает 6 Гбайт. Этого достаточно, чтобы без проблем работать одновременно в 6–7 дистрибутивах Linux.

Жесткий диск. Большинство систем виртуализации сохраняют файловые системы гостей в большом файле в системе хозяина. Таким образом, гости получают доступ к файлам жесткого диска не прямо, а опосредованно, через систему виртуализации. Следовательно, доступ к файлам в «гостевой» системе осуществляется значительно медленнее, чем в системе хозяина, в 2–3 раза.

CD/DVD-приводы. CD- и DVD-приводы выделяются хозяином гостям. В любом случае предоставляется доступ «только для чтения». Мне не известна ни одна система виртуализации, которая позволяла бы записывать CD и DVD в гостевой системе.

Большинство программ виртуализации дают возможность присвоить каждому виртуальному CD или DVD ISO-файл. Тогда гость вместо того, чтобы пользоваться реальным приводом, обращается к такому файлу. Это исключительно полезно в тех случаях, когда необходимо многократно устанавливать одни и те же программы. При необходимости вы можете без особого труда сами извлечь ISO-файл с CD или DVD.

```
root# dd if=/dev/scd0 of=файл.iso bs=2048
```

Графический адаптер. Для более или менее эффективного использования графических возможностей на каждой гостевой системе необходимо установить специальный драйвер, настроенный на виртуализационное ПО хозяина. В зависимости от применяемой системы виртуализации существуют определенные ограничения в области использования трехмерной графики.

Звуковые функции. Большинство программ виртуализации предоставляют гостевой системе виртуальную звуковую карту и перенаправляют звуковой вывод на аудиосистему хозяина. Если вы не выдвигаете экстраординарных требований к аудиосистеме (например, вам не требуется эффект «звук вокруг»), то такого механизма вполне достаточно.

USB-устройства и внешнее аппаратное обеспечение. Ввод, осуществляемый с помощью клавиатуры и мыши, направляется из системы-хозяина в систему-гость. От применяемой системы виртуализации зависит, к каким внешним устройствам будут иметь доступ пользователи гостевых машин. USB-устройства, к сожалению, поддерживаются не всеми системами виртуализации, а если и поддерживаются, то с серьезными ограничениями.

Виртуальные машины и проблемы сетевых соединений

Система виртуализации позволяет гостевым машинам пользоваться сетевой структурой машины-хозяина с помощью виртуальной сетевой карты. Существуют различные методы перенаправления трафика, проводимого через виртуальную сетевую карту, в реальную сеть. Можно задать несколько виртуальных сетевых адаптеров, сообщающихся различными методами (подобно тому, как настоящий компьютер может иметь несколько сетевых адаптеров, например для LAN и WLAN).

В дальнейшем мы будем пользоваться номенклатурой программы VirtualBox. Учитывайте, что некоторым программам известны не все возможные варианты виртуализации либо определенные варианты могут порой иметь альтернативные малоупотребительные названия.

- **Сетевой мост** — при использовании механизма сетевого моста (bridged networking) гость является дополнительным клиентом в локальной сети. Этот вариант является идеальным на тот случай, если в сети (но не на машине-хо-

зяине) присутствует DHCP-сервер либо если машина-хозяин подсоединена к ADSL- или WLAN-маршрутизатору. В таком случае сетевые конфигурации сообщаются гостевым машинам через подобный сервер или маршрутизатор и гостевые машины получают доступ как к локальной сети, так и к Интернету. Если компьютер-хозяин оборудован несколькими сетевыми интерфейсами, то следует указать, какой из них будет применяться для создания сетевого моста (через этот интерфейс будет обеспечиваться соединение с Интернетом).

- **NAT** — при использовании механизма NAT (преобразование сетевых адресов) сама система виртуализации работает по отношению к своим гостевым машинам как DHCP-сервер и выполняет функции маскарадинга (см. также раздел 19.3). Таким образом, гостевые машины могут использовать соединение с Интернетом, настроенное на машине-хозяине. Доступ к локальной сети в таком случае становится невозможным по причине несовпадения адресных пространств локальной сети и сети NAT системы виртуализации.
- **Сетевая модель «Только хозяин»** — при использовании такого варианта гость может обмениваться через сеть информацией только с хозяином, но не с другими компьютерами, подключенными к данной сети или Интернету. Целесообразно использовать такой вариант в тех случаях, когда вы хотите построить на нескольких виртуальных машинах испытательную систему, недоступную извне.
- **Внутренняя сеть** — программа виртуализации создает виртуальную сеть, по которой могут сообщаться только виртуальные машины. При использовании такого варианта виртуальные машины не имеют доступа ни к локальной сети, ни к Интернету.

Обмен данными между хозяином и гостем

При построении сетей действует следующий принцип: машина-гость не должна иметь прямого доступа к жесткому диску, которым непосредственно пользуется машина-хозяин. Две операционные системы не могут одновременно управлять контроллером одного жесткого диска, так как в противном случае неизбежны потери данных. По этой причине программа виртуализации имитирует на машине-хозяине специальный жесткий диск для машины-гостя и отвечает за то, чтобы такие виртуальные жесткие диски отображались в файловой системе машины-хозяина. Однако это означает, что машине-хозяину закрыт непосредственный доступ к файловой системе машины-гостя и наоборот.

Сетевые каталоги. Следовательно, наиболее быстрый путь обмена данными лежит через сетевые каталоги. Проще всего сделать так: на определенном компьютере, к которому обеспечивается доступ через сеть, как для машины-хозяина, так и для машин-гостей, запускается сервер NFS или Samba. В составе некоторых виртуализационных программ содержатся подобные функции (*совместно используемые каталоги* и т. д.). Однако, по моему мнению, выгода от использования такого принципа невелика, и конфигурация иногда излишне усложняется без надобности.

Область обмена данными (буфер обмена). Многие программы виртуализации позволяют обмениваться выделенным текстом через область обмена данными. К сожалению, такая система обычно далека от совершенства и работает с ошибками. Виновата в этом так называемая графическая система X, которая должна отличать буфер обмена, созданный специально для данного фрагмента выделенного текста, и буфер обмена, который создан для текста, скопированного с помощью сочетания клавиш Ctrl+C.

Иногда непонятно, какой из этих областей обмена данных пользуется система виртуализации. Кроме того, функциональность буфера обмена ограничивается копированием только текстовой информации. Если вам потребуется быстро скопировать диаграмму Excel с гостевой машины на машину-хозяина, на которой открыт документ, созданный в редакторе OpenOffice, то вы будете разочарованы.

Программы для виртуализации

Спектр предлагаемых инструментов виртуализации, как в коммерческом сегменте, так и среди свободно распространяемого ПО, необозримо велик. В следующем списке кратко рассмотрены важнейшие флагманы рынка виртуализации. В скобках указано, является ли данный продукт коммерческим или распространяется свободно, какая фирма занимается разработкой и реализует на рынке соответствующую продукцию.

- **VMware (коммерческий, EMC).** Фирма VMware — бесспорный лидер на рынке программ для виртуализации. Список производимой ею продукции начинается с пользовательских программ для ПК (рабочая станция и проигрыватель VMware) и заканчивается рядом мощных программ для сервера (VMware Server, ESXi, vSphere). Отдельные программы распространяются бесплатно, но не с открытым кодом. В качестве системы-хозяина поддерживаются Windows, Linux, а в отдельных случаях и Mac OS X. Некоторые продукты VMware работают вообще без операционной системы, «с нуля» (bare metal).

  Я имею богатый опыт использования станции VMware, которую много лет применял в операционных системах Windows и Linux. В итоге к 2008 году накопилось столько проблем с сетью и работой с клавиатуры, что я заменил имевшиеся у меня установки VMware на VirtualBox (образы дисков VMware можно продолжать использовать в VirtualBox, однако виртуальную машину потребуется настроить заново).

- **VirtualBox (частично бесплатная программа, Sun/Oracle).** Функции программы VirtualBox в целом похожи на функции рабочей станции VMware, таким образом VirtualBox также подходит для настольной виртуализации. В качестве системы-хозяина поддерживаются Windows, Linux и Mac OS X. Для частных пользователей программа VirtualBox бесплатна; кроме того, есть свободно распространяемая версия этой программы, которая может использоваться и коммерческим образом, на условиях стандартной общественной лицензии (GPL). VirtualBox исключительно быстро развивалась в последние годы. Такой факт, что за год выходило несколько версий программы, говорит

о том, что VirtualBox хорошо совместим с новейшими версиями ядра и X-версиями.

- **KVM/QEMU (свободно распространяемая программа, Red Hat).** Собственно, KVM — это просто модуль ядра, который радикально ускоряет работу эмулятора QEMU при использовании современных процессоров, при том что раньше этот эмулятор работал достаточно медленно. С тех пор как KVM официально вошел в состав ядра, а Red Hat купил Qumranet — фирму, разработавшую KVM, — значение модуля KVM резко выросло и он уже считается стандартным виртуализационным решением в дистрибутивах Fedora, Ubuntu и, конечно же, для версии 6 Red Hat Enterprise Linux. KVM одинаково хорош для применения как на ПК, так и на сервере. И все же по таким показателям, как понятность для пользователей, совместимость и скорость, KVM пока не может конкурировать с аналогичными коммерческими программами — Vmware, VirtualBox и Xen. В качестве системы-хозяина поддерживается только Linux.
- **Xen (частично бесплатная программа, Citrix).** Xen — это гипервизор, функционирующий без операционной системы. Виртуализированные гостевые системы работают в так называемых доменах (domU), причем первый домен имеет особые привилегии и в определенном смысле сравним с системой-хозяином в других программах виртуализации. Во многих практических ситуациях Xen значительно эффективнее, чем другие системы виртуализации. Однако в то же время настройка и конфигурирование гостевых систем (доменов) требует гораздо больших усилий. Это не в последнюю очередь объясняется тем, что расширения ядра, необходимые для правильной работы Xen, очень объемны и, несмотря на все приложенные усилия, пока не входят в состав официальной версии ядра. Если же вы готовы вложить в работу с Xen много времени, то достигнете выдающихся результатов, но для использования от случая к случаю Xen не годится.
- **OpenVZ и Virtuozzo (частично бесплатные программы, Parallels), а также Linux-VServer (свободно распространяемое ПО).** OpenVZ, базирующийся на его основе коммерческий продукт Virtuozzo и технически сходное виртуализационное решение Linux-VServer позволяют обустраивать много изолированных сред в одном дистрибутиве Linux. OpenVZ или Virtuozzo при работе исходят из того, что в системах «хозяина» и его «гостей» работает одна и та же версия Linux. Эта концепция отлично подходит для тех случаев, когда необходимо виртуализировать несколько (много!) аналогичных серверов. Такая система частично используется провайдерами интернет-хостинга, которые предлагают недорогие виртуальные корневые серверы.
- **Hyper-V (коммерческая программа, Microsoft).** Корпорация Microsoft поначалу не успела поучаствовать в разделе рынка виртуализации, но сейчас прилагает титанические усилия, чтобы сделать собственное виртуализационное решение — Hyper-V — конкурентоспособным. Hyper-V воспринимает систему Windows Server как систему-хозяина, но при этом может поддерживать Linux в качестве гостевой системы. Компания даже разработала для этой цели

собственные драйверы ядра Linux, причем эти драйверы с открытым кодом (такой шаг дался Microsoft с большим трудом, так как эта корпорация очень долго представляла Стандартную Общественную Лицензию в самом черном свете).

Немного из собственного опыта. Возможно, вы зададитесь вопросом: «Насколько активно использовалась виртуализация и соответствующие программы при написании этой книги?» Отвечу просто и кратко: «Активнее, чем когда-либо ранее!» Из всех перечисленных программ мне ближе всего VirtualBox. Я многократно устанавливал с помощью этой программы почти все дистрибутивы, которые упоминаются в книге (32/64 бит, с Рабочим столом от KDE/Gnome и т. д.). Раньше приходилось одновременно применять до четырех компьютеров, чтобы параллельно тестировать несколько дистрибутивов без постоянных перезапусков, но в последние месяцы я работал с 6–7 окнами VirtualBox, открытыми одновременно. Это невероятно удобно! К тому же необыкновенно практичной оказывается возможность без малейшего риска испытывать тестовые версии на ранних этапах разработки.

Когда речь заходит о тестировании реального аппаратного обеспечения, то, разумеется, и возможности VirtualBox исчерпываются достаточно быстро. Информативные испытания скоростных характеристик, пробы современных аппаратных компонентов или параллельная установка нескольких версий Windows и Linux, как и ранее, проводятся только на реальном оборудовании. Поэтому все дистрибутивы, описанные в данной книге, тестировались не только виртуально, но и на реальных машинах. Роль испытательного полигона сыграли три ноутбука и два ПК.

В минувшие годы я очень активно использовал виртуализацию при работе серверами. Здесь я также отдал предпочтение свободной программе: KVM.

Межсистемная совместимость. К сожалению, обычно пользователи вынуждены исходить из того, что системы виртуализации несовместимы друг с другом. Гостевая система, установленная для работы с VMware, не может применяться с Xen (и наоборот). Такая несовместимость обусловлена двумя факторами: форматы виртуальных жестких дисков отличаются (при этом формат VMware принимается многими другими программами в качестве наименьшего общего знаменателя) и в зависимости от системы виртуализации в гостевой системе требуется установить различные дополнительные драйверы, расширения ядра и т. д.

Правда, коммерческие программы для виртуализации частично способны импортировать гостевые системы продуктов-конкурентов, но при этом опять же возможны проблемы. Сейчас предпринимаются попытки разработать единообразные форматы для гостевых систем и стандартизировать их. Когда это начинание увенчается успехом и увенчается ли вообще, остается только гадать.

Существует еще одно ограничение. Большинство систем виртуализации требуют установки в системе-хозяине определенных модулей или драйверов ядра. Поэтому, как правило, не удается параллельно эксплуатировать несколько систем виртуализации (например, одновременно задействовать машины с VirtualBox и VMware).

4.2. Установка VirtualBox (хост)

VirtualBox — это система виртуализации, которая используется на ПК и работает в Linux, Windows, Solaris и Mac OS X. В качестве гостевых систем поддерживаются практические все имеющиеся операционные системы из ряда x86 (в том числе Windows 7, Solaris и OpenBSD). VirtualBox является 64-бит-совместимой, может определять для гостевых систем несколько процессоров или ядер, поддерживает в гостевой системе 3D-функции (в достаточном объеме, чтобы пользоваться 3D-функциями в Linux, но без возможности применения полупрозрачных интерфейсов модели Aero Glass, используемой в Windows Vista и Windows 7), поддерживает моментальные снимки и т. д. Основные преимущества VirtualBox по сравнению с другими свободно распространяемыми программами для виртуализации (например KVM, Xen и т. д.) — это понятный и красивый пользовательский интерфейс и качественная организация расположения документов. Эти обстоятельства упрощают работу именно для начинающих.

В общем, функции VirtualBox и рабочей станции VMware схожи, но VirtualBox предоставляется частным пользователям бесплатно, а в некоторых урезанных вариантах даже имеет открытый код. Кроме того, система VirtualBox обеспечена технической поддержкой значительно лучше, чем рабочая станция VMware. Каждый год выходит несколько новых версий VirtualBox, причем не считая ежемесячных обновлений, поэтому можно быть уверенным, что VirtualBox будет совместима и с новейшими доступными на сегодняшний день версиями ядра и X-версиями.

Система VirtualBox сначала принадлежала компании InnoTek, которая разработала вместе с другой фирмой (Connetix) продукт Virtual PC. Корпорация Microsoft приобрела эту виртуализационную программу в 2003 году и по-прежнему предлагает Virtual PC, однако официально поддерживает только виртуализацию для систем Microsoft. С 2004 года VirtualBox и Virtual PC развиваются независимо друг от друга. В феврале InnoTek вошла в состав Sun, и сразу после этого Sun стала частью Oracle. В данном разделе мы поговорим о VirtualBox версии 4.1. Актуальная информация и подробная документация об этой системе находится по адресу www.virtualbox.org.

Проблемы. Разработчики ядра немало натерпелись с VirtualBox: сама программа VirtualBox и ее модули для ядра считаются неаккуратно написанными и полны ошибок: именно они вызывают бесчисленные сбои ядра и соответствующие сообщения об ошибках. Некоторые разработчики ядра даже называют драйверы ядра для VirtualBox «гнилой лажей» (tainted crap) и просто не гарантируют никакой поддержки, если загружаются модули ядра для VirtualBox. Больше информации вы найдете по сайтах http://www.phoronix.com/vr.php?view=OTk5Mw и https://lkml.org/lkml/2011/10/6/317.

Установка VirtualBox в Linux. В большинстве дистрибутивов предлагаются готовые пакеты программы VirtualBox. В Fedora потребуется обратиться к версии источника пакетов rpmfusion. Установка совершенно проста.

В системе-хозяине VirtualBox обращается к трем модулям ядра: `vboxdrv`, `vboxnetadp` и `vboxnetfit`. В некоторых дистрибутивах эти модули также поставляются вместе

в виде пакета, который обновляется при каждом обновлении ядра. Если такой механизм не задействован, то модули ядра должны заново компилироваться с помощью следующей команды после установки и каждого обновления ядра (VirtualBox указывает на эту команду при каждом запуске виртуальной машины, когда система обнаруживает, что при загрузке в наличии нет модулей ядра):

```
root# /etc/init.d/vboxdrv setup
```

Исходный код для модулей ядра устанавливается вместе с VirtualBox. Однако для компиляции также необходимы C-компиляторы `gcc` и файлы заголовков ядра. В Ubuntu эти требования выполняются по умолчанию, в других дистрибутивах требуется устанавливать соответствующие пакеты.

Пакеты VirtualBox от компании Oracle. Вместо пакетов для VirtualBox, которые поставляются вместе с вашим дистрибутивом, вы также можете установить версию пакетов от Oracle. Это целесообразно, в первую очередь, в случаях, когда Oracle предлагает более новую версию VirtualBox, нежели ваш дистрибутив.

На сайте http://www.virtualbox.org/wiki/Linux_Downloads можно скачать VirtualBox в различных форматах: в виде пакетов RPM или Debian для различных дистрибутивов, а также в виде универсального установщика, который запускается следующим образом:

```
root# chmod u+x VirtualBox_n.run install
root# ./VirtualBox_n.run install
```

Вам в любом случае придется самостоятельно компилировать модули ядра с помощью команды:

```
root# /etc/init.d/vboxdrv setup
```

По возможности следует установить для Virtualbox пакет `dkms`. В таком случае DKMS управляет модулями VirtualBox и при обновлении ядра автоматически совершает новую компиляцию. Однако во многих инсталляциях VirtualBox этот механизм не всегда работает надежно.

Источник пакетов Oracle-APT. Для пользователей Ubuntu и Debian существует собственный источник пакетов APT, обеспечивающий автоматические обновления в рамках выбранной основной версии. Для этого необходимо добавить к пути `/etc/apt/sources.list` одну из следующих строк:

```
deb http://download.virtualbox.org/virtualbox/debian precise contrib
deb http://download.virtualbox.org/virtualbox/debian oneiric contrib
```

Кроме того, нужно выполнить обе следующие команды, чтобы установить ключ к источнику пакетов.

```
root# wget -q http://download.virtualbox.org/virtualbox/debian/sun_vbox.asc
root# apt-key add sun_vbox.asc
```

Установите также VirtualBox с помощью команды `apt-get` или `aptitude`:

```
root# apt-get update
root# apt-get install virtualbox-4.1
```

Источник пакетов Oracle-Yum. Пользователи дистрибутивов, совместимых с Yum (Fedora, openSUSE и др.), могут настроить источник пакетов Yum:

```
root# wget -q http://download.virtualbox.org/virtualbox/debian/sun_vbox.asc
root# rpm --import sun_vbox.asc
```

Кроме того, скачайте соответствующий файл *.repo с сайта VirtualBox и скопируйте его в каталог /etc/yum.repos.d:

```
[virtualbox]
name=VirtualBox
baseurl=http://download.virtualbox.org/virtualbox/rpm/fedora/$releasever
enabled=1
gpgcheck=1
```

Теперь установка будет осуществляться с помощью команд install или zypper install.

Подготовительные работы. При установке VirtualBox настраивается группа vboxusers. Только пользователи, которые входят в эту группу и обладают правом внесения изменений в файл устройства /dev/vboxdrv, могут запускать виртуальные машины. Следовательно, при первом запуске VirtualBox вам следует указать свою учетную запись из группы vboxusers и дополнительно выйти из системы и снова войти в нее. При этом замените имя kofler своим логином.

```
root# usermod -a -G vboxusers kofler     (Fedora, Debian, Ubuntu и т. д.)
root# groupmod -A kofler vboxusers       (openSUSE, SUSE, Novell)
```

В будущем всякий раз, как вы будете запускать систему-хозяина, будет выполняться сценарий Init-V-Script /etc/init.d/vboxdrv. Он загружает одноименный модуль ядра и обеспечивает выполнение основных предпосылок для эксплуатации VirtualBox.

Кроме того, через меню KDE или Gnome запускается пользовательский интерфейс VirtualBox, и эту операцию также можно выполнить с помощью команды VirtualBox. Система VirtualBox сохраняет настройки для виртуальных машин и виртуальных жестких дисков в каталоге ~/.VirtualBox. Если не хотите помещать эти файлы здесь, можете указать другие пути, выполнив команду Файл ▸ Глобальные настройки.

Установка для Windows и Mac OS X. Для частных пользователей установка VirtualBox в актуальных версиях Windows и Mac OS X не составляет никакого труда. Нужно просто скачать актуальную версию с сайта virtualbox.org, выполнить установку и перезапустить Windows.

Пакет расширений. На сайте Oracle можно скачать так называемый пакет расширений (extension pack). После установки VirtualBox этот пакет можно выполнить, и в VirtualBox появятся дополнительные функции. В частности, вы сможете обращаться с виртуальных машин к USB-устройствам и iSCSI-серверу, а также управлять виртуальными машинами по протоколу RDP (протокол удаленного дисплея) с другого компьютера, работающего в сети. Такие расширения распространяются только в двоичной форме, то есть речь не идет об открытом коде. Для коммерческого использования этих расширений требуется специальная лицензия от Oracle!

4.3. Настройка машины с VirtualBox (гость)

Настройка виртуальной машины в Linux

В этом разделе рассказывается, как настроить в рамках VirtualBox виртуальную машину в Linux. При этом неважно, в какой операционной системе будет использоваться VirtualBox: в Windows, Linux или Mac OS X.

При настройке виртуальной машины вспомогательные функции выполняет специальный ассистент. После выбора типа операционной системы (например, Linux с ядром 2.6.*n*, но данная настройка также действует для версий ядра 3.*n*), желаемого объема оперативной памяти и настройки виртуального жесткого диска VirtualBox покажет обобщенную информацию по аппаратным компонентам машины. Здесь при необходимости можно выполнять дальнейшие настройки, например изменить параметры доступа к сети или указать в качестве источника данных для CD/DVD-привода файл ISO.

Закончив конфигурацию, запустите виртуальную машину. VirtualBox показывает каждую виртуальную машину в отдельном окне. В этом окне можно установить Linux, как на реальном компьютере.

Сетевое соединение между хозяином и гостем по умолчанию осуществляется через NAT. При этом гостевая машина хотя и имеет соединение с Интернетом, однако не может обмениваться никакими данными по локальной сети. В подразделе «Виртуальные машины и проблемы сетевых соединений» раздела 4.1 описаны другие сетевые варианты, которые поддерживаются в VirtualBox. Я использовал такой вариант настройки, при котором машина-хозяин была соединена с маршрутизатором ADSL. Для сетевой конфигурации виртуальных машин я применил сетевой мост. С помощью такого моста все виртуальные машины становятся элементами локальной сети, благодаря чему упрощается обмен данными между системой-хозяином и гостевыми системами (SSH, NFS, Samba и др.).

Виртуальная машина автоматически оказывается в фокусе для работы с мышью и клавиатурой, когда вы нажимаете определенную клавишу. По умолчанию окно выводится из фокуса нажатием клавиши Правый Ctrl. В основном окне VirtualBox, которое открывается при выполнении команды Файл ▸ Глобальные настройки ▸ Ввод, можно задать и другую хост-клавишу. К сожалению, сочетания клавиш здесь использовать нельзя. В табл. 4.1 представлены различные сочетания хост-клавиш. Еще некоторые сочетания клавиш приведены в меню Машина окна VirtualBox.

Таблица 4.1. Сочетания клавиш в VirtualBox

Сочетание клавиш	Значение
Хост-клавиша+F	(Де)активировать полноэкранный режим
Хост-клавиша+Delete	Отправить Ctrl+Alt+Delete в гостевую систему
Хост-клавиша+Backspace	Ctrl+Alt+Backspace в гостевую систему
Хост-клавиша+F*n*	Отправить Ctrl+Alt+Fn в гостевую систему
Хост-клавиша+S	Сохранить текущее состояние виртуальной машины (мгновенный снимок)
Хост-клавиша+H	Выключить виртуальную машину через ACPI
Хост-клавиша+R	Мгновенное выключение виртуальной машины (внимание: сброс!)

Установка расширений для гостевых машин. После того как установка будет завершена, на гостевую машину еще потребуется установить так называемые *гостевые дополнения*. Вы предоставляете гостевой системе дополнительные драйверы, улучшая таким образом взаимодействие между ней и хозяином. Теперь указатель мыши можно вывести из гостевой машины, а виртуальное разрешение ее экрана автоматически подстраивается под размер окна (!), обмен данными между гостем и хозяином теперь может происходить через совместно используемые каталоги, можно копировать текст из буфера обмена и т. д.

В некоторых дистрибутивах предоставляются готовые пакеты с гостевыми расширениями:

```
openSUSE: virtualbox-ose-guest-tools, xorg-x11-driver-virtualbox-ose
Ubuntu:   virtualbox-ose-guest-utils, virtualbox-ose-guest-x11, virtualbox-ose-guest-
dkms
```

В других дистрибутивах в случаях, когда требуется новейшая версия гостевого расширения, это расширение требуется установить вручную. Для инсталляции вставьте требуемый CD/DVD и выполните команду Устройства ▸ Установить гостевые расширения. Интегрировав компакт-диск с гостевыми расширениями в файловую систему виртуальной машины (как правило, это делается щелчком на значке CD, иногда — с помощью команды `mount`, например `mount/dev/sr0/media/cdrom`), выполните следующую команду:

```
root# sh /media/cdrom/autorun.sh
```

СОВЕТ

При тестировании очень часто виртуальная машина не может опознать CD с гостевыми расширениями. В таком случае лучше останавливать виртуальную машину перед установкой гостевых расширений и выбирать в окне конфигурации привода CD/DVD ISO-образ гостевых расширений. При использовании такого метода после перезапуска виртуальной машины не возникает никаких проблем с доступом к диску с гостевыми расширениями.

Теперь программа установки инсталлирует три новых модуля ядра: `vboxadd`, `vboxvideo` и `vboxvfs`, а также новый X-драйвер, и при следующем запуске виртуальной машины этот драйвер уже будет использоваться.

В Ubuntu установка гостевых расширений запускается автоматически. В большинстве других дистрибутивов Linux необходимо сначала установить различные пакеты, содержащие C-компилятор и файлы заголовков ядра. Сначала выполните обновление, чтобы гарантировать, что установленная версия ядра и файлы заголовков ядра будут соответствовать друг другу! Чтобы узнать, какая версия ядра используется у вас на машине, выполните команду `uname -a`.

```
root# aptitude install gcc make linux-headers-n.n-plattform (Debian)
root# yum install gcc make kernel-headers kernel-devel      (Fedora)
root# zypper install gcc make kernel-source kernel-syms     (openSUSE)
```

При работе с современными версиями Fedora будьте внимательны: есть две разные версии ядра, и только в одной из них поддерживается PAE. Вы можете узнать, какой именно версией пользуетесь, с помощью команды `uname -r`. Если в результате получается последовательность символов, содержащая `pae`, то ядро

работает с поддержкой PAE. В таком случае необходимо установить `kernel-devel` как пакет `kernel-PAE-devel`. Только тогда будет возможно скомпилировать совместимый модуль на работающем ядре.

Установка виртуальной машины в Windows

Если у вас есть установочный диск, соответствующий файл ISO, а также действующая лицензия и ключ к ней, то вы можете установить VirtualBox и в системе Windows. Во время проведенных мной испытаний установка VirtualBox с гостевыми расширениями в Windows XP и Windows 7 прошла без проблем (рис. 4.1).

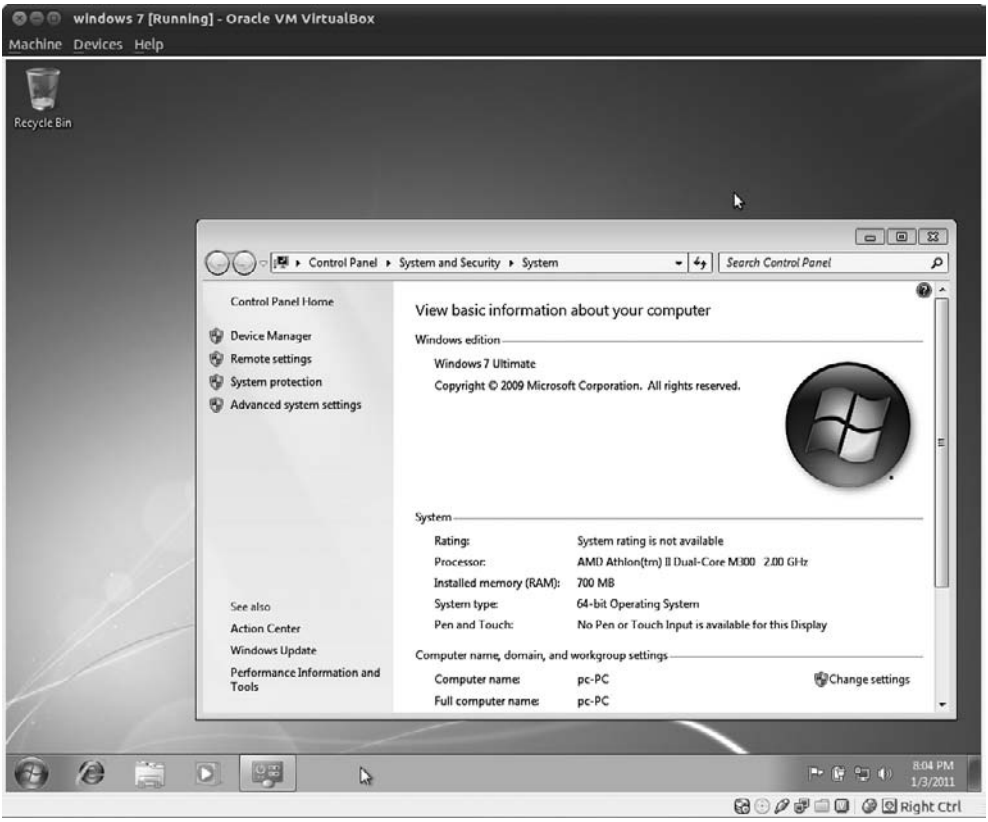


Рис. 4.1. 64-битная версия Windows 7 RC в виртуальной машине VirtualBox

VirtualBox сразу же начинает взаимодействовать со всеми ходовыми версиями Windows (в том числе с Windows 7, 64- и 32-битной версиями). Правда, в Windows 7 и в Windows Vista трехмерные эффекты Рабочего стола с полупрозрачным размытием (Aero Glass) остаются недоступны. Из соображений повышения эффективности рекомендуется работать с Windows XP, если вам не нужны особые функции, имеющиеся в Windows 7 и Windows Vista.

Повремените с онлайн-регистрацией до тех пор, пока производительность системы не будет вас устраивать. Если позже при настройке виртуальной машины вы увеличите оперативную память или измените другие параметры виртуального аппаратного обеспечения, то регистрацию придется повторить!

Дополнительные функции VirtualBox

Организация совместно используемого каталога. Для облегчения обмена данными между гостями и хозяином можно создать на машине-хозяине так называемый *каталог обмена*. Он работает в конкретной виртуальной машине. Во время создания этого каталога виртуальная машина должна быть остановлена. Для проведения настройки нажмите кнопку Изменить, в результате чего откроется окно настройки. Перейдите на вкладку Общие каталоги и выберите в системе-хозяине локальный каталог, а потом дайте ему имя (например, myshare).

В гостевой системе потребуется вручную выполнить команду `mount`, чтобы получить доступ к общему каталогу. При этом имя myshare потребуется заменить тем именем, которым вы пользовались в процессе конфигурации.

```
root@gast# mkdir /media/vbox-share
root@gast# mount -t vboxsf myshare /media/vbox-share
```

Каталоги обмена предусмотрены и в том случае, если гостевой системой является Windows и при условии, что в этой системе установлены гостевые расширения. Запустите Проводник Windows, а затем выполните команду Дополнительно ▸ Установить соединение с сетевым диском. Доступ к общему каталогу будет осуществляться через сетевой каталог `\\vboxsrv\myshare`.

Использование USB-устройств в виртуальных машинах. Если вы установили пакет с расширениями (Extension Pack) для VirtualBox, то можете работать в виртуальных машинах и с USB-устройствами. Эта функция работает только в том случае, если в системе-хозяине *не* применяется USB-устройство. Обычно в системе-хозяине USB-носитель автоматически включается в файловую систему. В таком случае вам потребуется вновь удалить USB-устройство из системы.

Еще одно необходимое условие заключается в том, что пользователь, работающий с VirtualBox, должен входить в группу vboxusers. Наконец, следите за тем, чтобы контроллер USB 2.0 на вкладке окна USB был активизирован при настройке виртуальной машины. В данном окне можно также задать фильтр, который будет присваивать USB-устройство определенной виртуальной машине. Правда, последнее требование не является обязательным. Кроме того, после включения USB-устройства его можно динамически присвоить статусной строке VirtualBox с помощью значка USB.

Однако когда я испытывал на виртуальных машинах устройства, подключаемые через USB (сканер и цифровой фотоаппарат), они безупречно работали в виртуальных машинах, может, только чуть медленнее, чем в системе-хозяине. Измерения скорости при этом не проводились.

Импорт и экспорт виртуальных машин. Чтобы перенести виртуальную машину на другой компьютер, нужно выполнить в меню команду Файл ▸ Экспорт модуля.

Так создается *виртуальный модуль*, то есть виртуальная машина, предназначенная для переноса данных и состоящая, как правило, из двух файлов: в OVF-файле содержится описание виртуальной машины, а в VMDK-файле — архивированный образ жесткого диска. Такую виртуальную машину можно снова установить с помощью команды меню Файл ▸ Импорт модуля при инсталляции другого экземпляра VirtualBox. Хотя формат для виртуального модуля и является стандартизированным, переход с Vmware на VirtualBox и наоборот, к сожалению, невозможен. Причина заключается в различиях виртуального аппаратного обеспечения, применяемого в каждой из систем.

5 Окна терминалов и работа с консолью

До сих пор мы рассматривали Linux как настольную систему. Мы познакомились с некоторыми программами из этой операционной системы. Данные приложения выглядят немного иначе, чем в Windows или Mac OS X, но, по сути, выполняют те же задачи и работа с ними не особенно отличается от двух вышеупомянутых систем. Однако на этом работа с Linux не заканчивается! Ведь есть еще одна сторона данной системы, которая на первый взгляд может показаться пугающей.

Опытные пользователи Linux выполняют команды в текстовой консоли или в окне консоли и получают ответ также в текстовом виде. Мышь играет лишь второстепенную роль, графические пользовательские интерфейсы остались не у дел.

Однажды научившись работать с окном терминала, очень многие задачи вы сможете решать гораздо эффективнее. Можно связывать друг с другом команды Linux, запускать одну программу на фоне другой, автоматически выполнять команды, автоматизировать работу маленьких программ (сценариев). Все эти возможности будут у вас не только тогда, когда вы работаете с локальным компьютером, но и тогда, когда подключаетесь к компьютеру удаленно через сеть.

Разумеется, обычные офисные работники гораздо реже пользуются консолью, нежели программисты и системные администраторы. В любом случае работа с нею входит в «джентльменский набор» любого пользователя, который на самом деле желает научиться работать с Linux. Важность этих навыков станет особенно очевидна тогда, когда графическая система не заработает из-за ошибочной конфигурации или если вам вдруг понадобится администрировать удаленный корневой сервер.

В этой главе будет сделан лишь вводный обзор способов работы с терминалами, которые также называются консолями. За выполнение программ в окне терминала отвечает так называемая оболочка (shell). В Linux на выбор предоставляется несколько оболочек. Чаще всего применяется оболочка bash, которой посвящена следующая глава книги.

В других главах различные команды Linux будут рассмотрены более подробно. Эти команды служат, например, для управления файловыми системами (`ls`, `cp`, `mv`, `ln`, `rm`) для поиска файлов (`find`, `grep`, `locate`), для управления сетевыми функциями (`ping`, `ifconfig`, `ssh`) и т. д. Кроме того, мы подробнее изучим многие особенности Linux.

5.1. Текстовые консоли и окна консолей

Текстовые консоли. С системой Windows можно работать только в графическом режиме. При использовании Linux вы можете применять еще и так называемые текстовые консоли. В большинстве дистрибутивов имеется по шесть текстовых консолей. Переход между ними осуществляется с помощью сочетания Alt+F1 для первой консоли, Alt+F2 для второй и т. д. Если компьютер уже работает с Linux в графическом режиме, то сочетание клавиш Ctrl+Alt+F1 выводит первую консоль, а Alt+F7 возвращает графический режим. В некоторых дистрибутивах (в том числе в Fedora) первая консоль зарезервирована для работы в графическом режиме.

Перед работой с текстовой консолью необходимо зарегистрироваться в системе. Когда вы закончите работу или захотите войти в систему под другим именем, нужно сначала выйти из системы. Для этого нажмите Ctrl+D (а не вводите новую команду).

На одной консоли можно запустить только одну команду (табл. 5.1). Пока эта команда выполняется, на другой консоли можно заняться еще чем-нибудь. Можно войти на одну консоль под именем `root` и заниматься администрированием, а на другую консоль войти под своим обычным логином и редактировать файл. Все консоли работают независимо друг от друга.

Таблица 5.1. Сочетания клавиш для активации текстовых консолей

Сочетание клавиш	Функция
Ctrl+Alt+F*n*	Переход из графического режима к работе с текстовой консолью *n*
Alt+F*n*	Переход от работы с текстовой консолью к работе с другой текстовой консолью *n*
Alt+F7	Переход обратно в графический режим (Alt+F1 в Fedora, Alt+F5 в Knoppix)
Alt+→/+←	Переход в предыдущую/следующую текстовую консоль
Shift+↑/↓	Листание вперед/назад
Ctrl+Alt+Delete	Завершить работу с Linux; только с текстовыми консолями, внимание: выполняется выключение системы!

С помощью Shift+↑ и Shift+↓ можно прокручивать вверх и вниз информацию, содержащуюся на экране текстовой консоли. Таким образом, можно повторно просмотреть результаты выполнения программ, которые уже скрылись из видимой области экрана.

Окно терминала (командное окно). Конечно же, излишне переходить из работы в графическом режиме к текстовой консоли только лишь для выполнения команд. Для этого вполне достаточно окна терминала. Такие окна еще называются командными, или окнами консоли.

В зависимости от применяемого дистрибутива и настольной системы предлагаются различные окна консолей, например `gnome-terminal` (Gnome), `konsole` (KDE) или `xterm` (X). Команда меню, предназначенная для запуска окна консоли, также отличается от дистрибутива к дистрибутиву. Рассмотрим несколько примеров:

- Fedora 17 (Gnome 3) — клавиша Windows ▸ Терминал;
- openSUSE 12.2 (KDE) — Программы ▸ Система ▸ Терминалы ▸ Программа терминала;

- RHEL 6 (Gnome 2) — Приложения ▸ Системные Инструменты ▸ Терминал
- Ubuntu 12.10 (Unity) — клавиша Windows ▸ Терминал.

В некоторых версиях Gnome окно терминала открывается еще удобнее — с помощью контекстного меню Рабочего стола. Для этого требуется установить дополнительный пакет `nautilus-open-terminal`.

Работа в окне терминала аналогична работе с текстовой консолью. Единственное отличие заключается в том, что во втором случае имеется полоса прокрутки, благодаря которой удобнее просматривать команды, выполненные некоторое время назад.

Важные сочетания клавиш. При работе с текстовыми консолями или окнами консолей часто могут пригодиться сочетания клавиш, предназначенные для быстрого ввода команд. В табл. 5.2 приведены важнейшие из них. Они работают только тогда, когда при конфигурации по умолчанию в качестве стандартной оболочки применяется bash — так обстоит дело в большинстве дистрибутивов. Если вы работаете с Gnome в окне консоли, то нужно выполнить команду Правка ▸ Комбинации клавиш и снять флажок Активировать все буквы горячих клавиш в меню.

Таблица 5.2. Сочетания клавиш для ввода команд в оболочке bash

Сочетание клавиш	Функция
Ctrl+A	Поставить курсор в начало строки (как Home)
Ctrl+C	Отменить программу
Ctrl+E	Поставить курсор в конец строки (как End)
Ctrl+K	Удалить строку, начиная от курсора
Ctrl+Y	Снова вставить только что удаленный текст
Ctrl+Z	Остановить программу (возобновить — с помощью fg или bg)
Tab	Автозавершение названия файла или команды
↑/↓	Листание для просмотра команд, введенных ранее

Огромное количество времени экономится благодаря автозавершению команд клавишей Tab. Кроме того, нажимайте Tab, когда имя файла уже можно однозначно предугадать — если возможность окончания только одна, система дополнит команду за вас. Если дважды нажать Tab, то откроется список всех файлов, названия которых начинаются с уже введенной последовательности букв.

Мышь. При работе с текстовыми консолями и окнами терминалов мышь играет второстепенную роль. Ее *невозможно* использовать для изменения положения курсора! Вся работа мыши ограничивается тем, что ее левой кнопкой можно копировать текст, а средней кнопкой — вставлять текст туда, где сейчас находится курсор.

Чтобы мышь работала в текстовых консолях (то есть *вне* графического интерфейса), необходимо запустить программу `gpm`.

Выполнение команд. Для выполнения команд в текстовой консоли или в командном окне нужно просто указать имя команды, иногда задать некоторые дополнительные параметры и нажать Enter. Команда `ls` выдает список файлов и подкаталогов, находящихся в данном каталоге.

```
user$ ls -l
-rw------- 1 user users 17708403 19. Mai 10:35 20060519_DN.pdf
-rw------- 1 user users   506614 29. Jun 12:11 angebot-katzbauer.pdf
drwxrwxr-x 3 user users     4096 13. Apr 11:31 bak
drwxrwxr-x 2 user users     4096 18. Jul 15:03 bin
-rw-r--r-- 1 user users   243571  3. Jul 09:14 DB20078.jpg
drwxr-xr-x 2 user users     4096  7. Apr 10:59 Desktop
...
```

Из данного примера видно, как в книге будет записываться ввод команды и результат ее выполнения: `user$` в начале первой строки означает, что команда выполнена обычным пользователем. Если бы в первой строке было указано `root#`, это бы означало, что команду выполнил администратор (в частности, это мог быть системный администратор). Выражения `user$` или `root#` являются вспомогательными индикаторами. Эти символы автоматически вводятся в каждую строку, вам их *не нужно* вводить! (Обычно требуется вводить только те символы, которые выделены полужирным.)

Возможно, на вашем компьютере вместо `user$` или `root#` будет выводиться другой текст, который часто содержит имя актуального каталога и/или компьютера. Для большей наглядности я буду опускать такие обозначения.

Иногда команда слишком длинная и ее не удается уместить в одну строку. В таких случаях она разбивается на несколько строк, разделяемых символом `\`. Такая команда может выглядеть следующим образом:

```
user$ gconftool-2  --set "/apps/panel/toplevels/top_panel_screen0/monitor" \
                   --type integer "0"
```

Итак, чтобы разбить команду на две строки, на месте разрыва строки нужно поставить `\`. Если же потребуется объединить команду в одну строку, то символ `\` необходимо убрать.

Выполнение команд на фоне работы программы. Команды можно выполнять и на фоне работы основной программы. Это означает, что вам не обязательно дожидаться окончания выполнения программы и вы можете работать далее. Тогда в конце строки, задающей команду, ставится символ `&`. Такой метод рекомендуется прежде всего, когда вы запускаете из консоли программу с графическим интерфейсом (например: `firefox &`).

Работа с правами администратора. В Linux нередко приходится работать в качестве администратора (например, с правами системного администратора). Даже если вы вошли в систему как обычный пользователь, существуют различные возможности выполнения команд, как если бы вы были администратором. Во многих дистрибутивах нужно всего лишь написать в открытой текстовой консоли или в окне консоли `su - l`. Так вы получаете права уровня `root` (разумеется, для этого необходимо знать пароль администратора). Теперь как администратор вы можете выполнять команды, предназначенные для работы с текстом. Если ввести команду `exit` или нажать Ctrl+D, вы снова переходите на уровень обычного пользователя (в Ubuntu в таком случае применяется команда `sudo`).

Советы о том, как перевести выполнение команды в фоновый режим при работе другой программы, как просмотреть список всех выполняемых команд (процессов) и т. д., даются в главе 8.

5.2. Просмотр и редактирование текстовых файлов

Команда less

В системах KDE или Gnome можно читать файлы прямо в файловом менеджере (Konqueror или Nautilus). Файл можно открыть щелчком правой кнопкой мыши в очень удобном редакторе. Если же вы работаете с текстовой консолью или с окном консоли, то для просмотра файлов лучше всего пользоваться командой `less`. Она может следовать и за другими командами, чтобы можно было спокойно прочитать результаты других команд — иногда очень длинные:

```
user$ less файл       (Постраничный показ файла)
user$ ls -l | less    (Постраничный показ каталога с файлами)
```

В табл. 5.3 приведены сочетания клавиш для работы с командой `less`.

Таблица 5.3. Клавиши и их сочетания для работы с командой less

Названия клавиш	Функция
Клавиши управления курсором	Перемещение текст вверх или вниз
Home, End	Переход к началу/концу текста
G, Shift+G	Переход к началу/концу текста
/ *образец* Enter	Поиск по направлению вперед
? *образец* Enter	Поиск по направлению назад
N	Повтор поиска вперед (next)
Shift+N	Повтор поиска назад
Q	Завершение (выход)
H	Отображение текста справки с другими обозначениями клавиш

В некоторых небольших системах, работающих с Linux (например, во встраиваемых устройствах, таких как сетевые жесткие диски NAS) команда `less` отсутствует. Иногда вместо нее установлена аналогичная команда `more`, которая использовалась ранее. В других случаях можно вывести с помощью команды `cat` все содержимое текстового файла (в таком виде не требуется перелистывание). Если вы хотите прочитать только первые строки — как, например, в файле лога — используйте команду `tail`.

СОВЕТ

Если вы показываете в текстовой консоли файл, в котором содержится не текст, а двоичные данные, то эти данные могут быть интерпретированы как специальные символы и в консоли возникнет путаница. В таком случае отобразятся только странные значки, то есть обработать и корректно представить набор символов не получится. Чтобы разобраться с этой ситуацией, выполните команду reset.

Препроцессор. В большинстве дистрибутивов `less` может показывать не только обычные текстовые, но и архивированные файлы, например в формате TAR. Для обеспечения этого препроцессор (программа первичной обработки данных)

анализирует данные, предназначенные для обработки, и передает результат команде `less`. В деталях этот процесс отличается от дистрибутива к дистрибутиву.

В Fedora и RedHat переменная среды `LESSOPEN` настроена так, что `less` сначала выполняет сценарий `/usr/bin/lessfile.sh` и показывает результат. В SUSE `less` выполняет сценарий `/usr/bin/lessopen.sh`.

В Debian и Ubuntu также установлены похожие сценарии или команды (`lessfile` и `lesspipe.sh`). Разница заключается в том, что `lesspipe` сразу же передает результаты работы `less`, а `lessfile` создает временный файл. Второй способ более медленный, но он имеет определенное преимущество — команде `less` сразу же становится известно количество строк и процентное положение в тексте. В Ubuntu по умолчанию активен сценарий `lesspipe`, а в Debian предусмотрена аналогичная строка в `~/.bashrc`, снабженная соответствующим комментарием.

Текстовые редакторы

В KDE или Gnome имеются программы `kate` и `gedit` — удобные текстовые редакторы с интуитивным управлением. Однако в текстовой консоли с этими программами работать нельзя — для них нужен редактор, работающий исключительно в текстовом режиме. В следующем разделе представлены наиболее популярные текстовые редакторы. Какой текстовый редактор установлен по умолчанию, зависит от дистрибутива.

Emacs, Jove, Jed, Jmacs. Особая роль среди программ-редакторов отводится GNUEmacs. В этом редакторе имеются невероятно широкие наборы функций, и такая программа с успехом может заменить программисту целую среду разработки. В табл. 5.4 приведены только простейшие команды. Эти же команды работают в редакторах Jove, Jed и Jmacs. При этом я говорю о наиболее упрощенной версии Emacs, совместимой только с самыми элементарными функциями.

Таблица 5.4. Сочетания клавиш для работы с Emacs

Сочетание клавиш	Функция
Ctrl+X, Ctrl+F	Загрузка нового файла
Ctrl+X, Ctrl+S	Сохранение данного файла
Ctrl+X, Ctrl+W	Сохранение файла под новым именем
Ctrl+G	Прерывание ввода команды
Ctrl+K	Удаление строки
Ctrl+X, U	Отмена удаления
Ctrl+X, Ctrl+C	Завершение работы Emacs (с запросом о сохранении)

Vi, Vim и Elvis. Одним из первоэлементов UNIX был редактор Vi, обычно представленный в Linux аналогичной программой Vim, которая совместима с этой операционной системой. Реже используется другой совместимый вариант — редактор Elvis. Оригинальный редактор Vi не входит состав Linux по причинам, связанным с авторским правом. Тем не менее, команду `vi` можно выполнять — она запускает Vim или Elvis.

В Vi имеется не меньше функций, чем в Emacs, однако научиться работать с ним сложнее. При этом Vi сравнительно компактен и обычно доступен в экст-

ренных системах. К тому же Vi есть практически в любых системах, созданных на основе UNIX. Программа является неофициальным стандартом в области UNIX/Linux, и многие другие программы автоматически вызывают ее для редактирования.

Важнейшее фундаментальное отличие от других редакторов заключается в том, что с Vi предусмотрено несколько режимов работы. Ввод текста осуществляется только в режиме вставки (табл. 5.5). Ввод большинства команд происходит в режиме complex command, который активируется символом : (табл. 5.6). Перед этим необходимо выйти из режима вставки, нажав клавишу Esc.

Таблица 5.5. Сочетания клавиш для работы с Vi

Сочетание клавиш	Функция
I	Переход в режим вставки
Esc	Выход из режима вставки
H/L	Движение курсора вправо/влево — функция работает и при использовании соответствующих клавиш управления курсором
J/K	Движение курсора вверх и вниз
X	Удаление символа
D D	Удаление данной строки
P	Вставка удаленной строки на позиции курсора
U	Общая отмена
:	Переход в режим complex command

Таблица 5.6. Сочетания клавиш для работы в режиме complex command

Сочетание клавиш	Функция
:w *имя*	Сохранение текста под новым именем
:wq	Сохранение результатов работы и выключение редактора Vi
:q!	Выключение редактора Vi без сохранения информации
:help	Вызов онлайн-справки

Joe. Это очень простой редактор. Сочетания клавиш аналогичны тем, что используются в программе для обработки текста Wordstar (табл. 5.7). Подробное описание всех команд выводится при выполнении на консоли команды `man joe`. Программа также иногда запускается под именами Jmacs или Jpico. При этом действуют другие сочетания клавиш, совместимые, соответственно, с Emacs или с Pico.

Таблица 5.7. Сочетания клавиш для работы с Joe

Сочетание клавиш	Функция
Ctrl+K, H	Открытие/закрытие окна справки
Ctrl+K, E	Загрузка нового файла
Ctrl+K, D	Сохранение файла (возможно сохранение под новым именем)
Ctrl+Y	Удаление строки
Ctrl+Shift+–	Отмена удаления
Ctrl+C	Завершение работы Joe (с запросом о необходимости сохранения информации)

Nano, Pico. Редактор Nano (или его вариант Pico) обладает небогатым набором команд, но зато прост в использовании. В этом редакторе в двух последних строках, выводимых на экране, дается обзор команд, которые имеются в вашем распоряжении. В большинстве современных дистрибутивов установлен только редактор Nano. Pico ранее был более широко распространен, однако его лицензия не полностью соответствовала принципам свободно распространяемого ПО и теперь этот редактор в Linux практически не применяется. Nano совместим с Pico, но не имеет проблем с лицензией.

Настройка текстового редактора по умолчанию. Некоторые программы сами запускают программу для редактирования файлов, по умолчанию в данном случае обычно используется Vi. Если вы хотите работать с другим редактором, нужно изменить переменные окружения `EDITOR` и `VISUAL` в `etc/profile` или `~/.profile`.

```
# Внесение дополнения в /etc/profile или ~/.profile
Export EDITOR=/usr/bin/jmacs
Export VISUAL=$EDITOR
```

5.3. Онлайн-справка

Команды `ls`, `cp` или `top`, которые обычно выполняются в окне консоли, не реагируют на нажатие клавиши F1. Кроме того, у них отсутствует меню Справка. Однако для них предусмотрены вспомогательные тексты, которые можно прочитать, введя различные команды.

- *команда* `-help` — употребляемая со многими другими командами, сообщает список всех параметров, используемых при работе с конкретной командой, а также краткое описание их значений.
- `man` *команда* — указывает текст `man`-справки для многих команд. Этот текст обычно многостраничный, его можно листать клавишами для управления курсора. Нажатие клавиши Q означает завершение работы справки.
- `help` *команда* — работает только с так называемыми командами оболочки, например `cd` или `alias`.
- `info` *команда* — это альтернатива для `man`. Такая система справки лучше всего подходит для просмотра крупных файлов справки. Как будет показываться текст справки — в виде `man` или `info`, зависит только от того, какую систему справки выберут разработчики программы. Однако `man` пользуется большей популярностью.

Команда man. Это команда для показа документации по большинству простейших команд, например `ls` или `cp`. Синтаксис: `man` *команда*, причем на месте *команда* указывается та команда, справку по которой необходимо просмотреть.

Можно задать диапазон (`man` *диапазон команда*), который ограничивает поиск текстов справки `man` определенной тематической областью. Например, если указать `3 printf`, будет выведен синтаксис функции `C printf`. Это ограничение полезно тогда, когда в различных тематических областях имеется несколько текстов `man` с одинаковыми наименованиями. В таком случае команда `man` показывает только первый найденный текст `man`.

Если вы хотите просмотреть все одноименные тексты man (из всех тематических областей), можно воспользоваться параметром -a. Когда вы прочитаете текст, закройте его, нажав клавишу Q, и увидите текст man по следующему разделу.

Во многих книгах по UNIX и Linux вместе с командами указываются и их названия в соответствии с man, например find(1). Таким образом, сразу понятно, как вызвать ту или иную команду. Обычно выделяются тематические области 1-9 и n; иногда команды из отдельных языков программирования распределяются по отдельным группам, и им присваиваются определенные буквы:

- 1 — пользовательские команды;
- 2 — системные вызовы;
- 3 — функции языка программирования C;
- 4 — форматы файлов, файлы-устройства;
- 5 — файлы конфигурации;
- 6 — игры;
- 7 — разное;
- 8 — команды для системного администрирования;
- 9 — функции ядра;
- n — новые команды.

Тексты файлов справки демонстрируются командой less, поэтому для навигации по тексту справки подходят и сочетания клавиш, обобщенные в табл. 5.3. Чтобы указать, из каких каталогов будут читаться тексты справки, нужно настроить переменную окружения MANPATH или придать управляющему файлу вид /etc/manpath.config.

В KDE и Gnome файлы справки можно читать в любом веб-обозревателе или электронном справочнике. На следующих примерах демонстрируется, как показать страницу man по команде ls и список всех страниц man:

```
user$ gnome-help man:ls
user$ khelpcenter man:ls
user$ khelpcenter 'man:(index)'
```

Команда help. Вызов справки по некоторым командам осуществляется не с помощью man, а с использованием help. Это касается всех команд, выполняемых прямо из оболочки. Оболочка — это интерпретатор команд, принимающий ваш ввод. Подробная информация по стандартной Linux-оболочке bash содержится в следующей главе.

Программа info. У справочных текстов есть недостаток — их очень сложно структурировать. В этом отношении возможности альтернативного формата info значительно шире, поэтому крупные справочные файлы обычно предоставляются именно в этом формате.

Программа info обычно вызывается в виде info *команда*. Если команда запускается без параметров, то программа показывает обзор доступных вспомогательных тем.

К сожалению, достоинство, связанное с четкой структурой, часто оборачивается недостатком: навигация по этим текстам оставляет желать лучшего, кроме того,

отсутствует механизм поиска, который позволял бы находить информацию в более широком диапазоне, чем страница, открытая в настоящий момент.

В табл. 5.8 приводится список клавиш для работы с info.

Таблица 5.8. Клавиши для работы с info

Названия клавиш	**Функция**
Пробел	Прокрутка текста по направлению вниз
Backspace	Прокрутка текста по направлению вверх
B, E	Переход к началу или концу данного раздела справки
Tab	Переход к следующей перекрестной ссылке
Enter	Переход к другой части справочной информации
N	Переход к следующей части справки на том же уровне иерархии
P	Переход к предыдущей части справки на том же уровне иерархии
U	Переход на один уровень иерархии выше
L	Возврат к последнему показанному фрагменту текста
H	Подробное практическое руководство
?	Обзор команд
Ctrl+O	Закрытие текущего подокна
Q	Завершение работы справки info

Вместо info можно также запустить редактор Emacs и открыть с помощью Alt+X info Enter или Ctrl+H, I режим справки. Здесь все перекрестные ссылки будут выделены цветом, по ним удобно переходить щелчком средней кнопки мыши. Удобная альтернатива для info — программа pinfo. В KDE и Gnome файлы справки любой системы лучше читать с помощью info.

6 Bash (оболочка)

В этой главе пойдет речь об оболочке bash. Эта программа позволяет выполнять команды в командном окне или текстовой консоли. Кроме того, оболочка является интерпретатором команд. В то же время bash располагает собственным языком программирования, применяемым для создания оболочковых программ (shell-сценариев).

В этой главе мы поговорим о bash как об интерпретаторе команд и как об инструменте для программирования. Наиболее важными темами, рассматриваемыми в главе, являются введение в работу с bash, переадресация ввода и вывода, обмен информацией между несколькими процессами (программные каналы, подстановка команд) и управление переменными оболочки. Если вы интересуетесь bash-программированием, обратите внимание на подборку важнейших элементов этого языка и разнообразные примеры. В конце главы приводится таблица, где перечислены специальные символы, используемые в bash.

6.1. Что такое оболочка?

Bash означает Bourne Again Shell. Это англоязычная игра слов: bash построена на основе оболочки Борна (bourne), которая, наряду с оболочкой Korn и C, считается одной из классических оболочек UNIX (again — с англ. «снова». — *Примеч. ред.*). В Linux можно работать и с двумя другими классическими оболочками, но по умолчанию обычно установлена оболочка bash.

Так что же такое оболочка? *Оболочка* в первую очередь предназначена для вызова команд и программ Linux. Таким образом, она является своего рода интерпретатором команд (сравнимым с *команда*`.com` из мира MS-DOS). Оболочка выполняется в любом командном окне, например `konsole`, `gnome-terminal`, `xterm`, и в любой текстовой консоли после входа в систему (логина).

В то же время оболочка является мощным языком программирования, с помощью которого можно автоматизировать рабочие процессы. Особые оболочковые команды позволяют использовать в рамках этой программы переменные, создавать запросы и циклы и т. д. Получаемые в результате программы, в зависимости от предпочтений автора, называются командными файлами, пакетными файлами, сценариями, процедурами оболочки и т. п. Независимо от названия, речь в данном

случае идет о простых текстовых файлах, которые выполняются (интерпретируются) оболочкой.

Версия. В этой главе описана bash-версия 4.*n*, но большая часть сообщаемой информации применима и к версии 3.*n*. Многие обновления версии 4.0 по умолчанию неактивны, и их нужно специально активировать (например, с помощью команды `shopt -s` *имя* в `/etc/bashrc`). Если вы не знаете, с какой версией оболочки работаете, выполните следующую команду:

```
user$ echo $0
-bash
user$ $BASH_VERSION
4.2.29(1)-release
```

Документация. В справке `man` оболочке bash посвящен большой раздел. Кроме того, есть объемный справочный материал, который выводится командой `info bash`. Разумеется, этот файл есть и в Интернете: http://www.gnu.org/software/bash/manual/bash.html.

Другие оболочки. Почти во всех дистрибутивах Linux bash считается стандартной оболочкой для работы с оболочками и окнами терминалов. Однако, воспользовавшись системой управления пакетами вашего дистрибутива, вы можете установить и многие другие оболочки. Профессионалы Linux особенно любят Z-оболочку zsh. Другие варианты — оболочки Korn (ksh или pdksh) и C (csh или tcsh). Чтобы опробовать любую из этих оболочек после установки, запустите командное окно и введите в него имя любой оболочки. Команда `exit` выведет вас обратно в предыдущую активную оболочку.

```
user$ zsh
hostname% ls   (Выполнение команд в zsh)
...
hostname% exit (Обратно в предыдущую оболочку)
user$
```

Выбор другой оболочки, загружаемой по умолчанию. Для любого пользователя, вошедшего в Linux, система предусматривает стандартную оболочку. Она автоматически запускается в командном окне при работе с терминалом, то есть стандартная оболочка сохраняется в файле `/etc/passwd`. Название оболочки указывается в самом конце строки с учетной записью каждого пользователя. Чтобы задать по умолчанию другую оболочку, выполните команду `chsh` (change shell). Программы оболочки сохраняются в каталоге `/bin`. Это значит, что требуется указать, например, `/bin/csh` в том случае, если вы собираетесь в дальнейшем работать с оболочкой C. Список доступных оболочек находится в `/etc/shells`.

/bin/sh. Большинство сценариев начинается с кода `#!/bin/sh`. Данная последовательность символов указывает, что сценарий должен выполняться оболочкой `/bin/sh`. Раньше в `/bin/sh` почти всегда ставилась ссылка на bash. Однако, поскольку bash (не в последнюю очередь благодаря множеству ее функций) считается относительно медленной и требует для работы большого количества памяти, в некоторых дистрибутивах вместо нее используется оболочка, лучше подходящая для выполнения сценариев. В Ubuntu применяется, например, оболочка Debian-Almquist (dash). Она быстра, но не полностью совместима с bash. Если при программирова-

нии вы используете bash-специфичные функции, нужно указать в первой строке #!/bin/bash.

```
user$ ls -l /bin/sh
.../bin/sh -> dash
```

6.2. Базовая конфигурация

Функциональные клавиши в bash. Конфигурация клавиатуры в bash глобально настраивается в файле /etc/inputrc с помощью ~/.inputrc. Если у вас не получается использовать специальные символы из национальных алфавитов либо если клавиши Delete, Home и End работают неправильно, нужно настроить inputrc так, как это показано далее. Все распространенные дистрибутивы по умолчанию конфигурируются именно так, но существует и множество других настроек.

```
# Файл /etc/inputrcbzw.~/.inputrc
setmeta-flagon
setconvert-metaoff
setoutput-metaon
"\e[1~":beginning-of-line
"\e[3~":delete-char
"\e[4~":end-of-line
```

Этот файл управляет функцией readline, применяемой внутри bash для обработки ввода с клавиатуры. Первые три команды позволяют опознать при наборе восемь символов битов, а также гарантируют, что эти символы ни в коем случае не будут преобразованы в другие символы, и, наконец, что эта информация действительно будет выведена. Последующие три строки управляют откликом на нажатие клавиш Delete, Home и End.

Изменения вступят в силу только после нового запуска оболочки. При работе с текстовой консолью нужно выйти из консоли и снова войти. В настольных системах необходимо открыть новое окно терминала.

Приглашения при вводе. В оболочке в начале каждой строки ввода в зависимости от дистрибутива отображается имя компьютера, имя пользователя и/или название текущего каталога. Последовательность символов обычно заканчивается на $, ~, > (у обычных пользователей) или на # (у администратора).

Основная конфигурация переменой окружения PS1, которая отвечает за внешний вид приглашения, обычно осуществляется в файле /etc/bash.bashrc, в RedHat и Fedora — в /etc/bashrc. Без применения конфигурации действует PS1="\s-\v\$". В этом случае bash отображает имя оболочки и номер версии. Чтобы индивидуально настроить PS1, измените файл ~/.profile. При вводе следующей строки в качестве приглашения просто отображается текущий каталог:

```
# Изменения в ~/.profile
PS1="\w\$"
```

Здесь \u является джокерным символом для имени пользователя, \h — для хостимени, \w — для всего актуального каталога, \W — для последней части актуального каталога и \$ (или #) — для завершения приглашения. Кроме того, с помощью

\[\e[0;nnm\] можно задавать цвет. Подробное описание конфигурации приглашений, а также список ANSI-кодов различных цветов находится в следующем справочном документе: http://tldp.org/HOWTO/Bash-Prompt-HOWTO/.

На моих компьютерах я использую следующую настройку:

```
PS1='\[\e[0;34m\]\u@\h:\W\$\[\e[0;39m\] '
```

В таком случае отображается голубое приглашение вида username@computername:folder. Кроме того, в приглашении указывается не весь путь к каталогу, а только его завершение. Так в случае с каталогом /usr/lib/nautilus имеем nautilus. Поэтому если вы находитесь в достаточно глубоком каталоге, при записи его адреса значительно экономится место.

В дополнение или в качестве альтернативы для PS1 можно также настроить переменную PROMPT_COMMAND. Она содержит команду, выполняемую всякий раз перед отображением PS1.

6.3. Ввод команд

Как правило, при работе в bash вводятся обычные команды. При этом bash облегчает вам работу благодаря применению множества удобных сочетаний клавиш. В частности, вы можете заново возвращаться к недавним командам с помощью клавиш управления курсором ↑ и ↓, что избавляет вас от набора большого объема информации. При выходе из оболочки последние введенные команды сохраняются в файле ~/.bash_history и, таким образом, вновь предоставляются вам при следующем входе в систему.

Строки с командами можно изменять в текстовом редакторе, то есть вставлять и удалять символы, где вам угодно. Раскладку клавиатуры в bash можно конфигурировать практически как угодно. В частности, на клавиатуре можно задавать специальные символы (а также клавиши управления курсором и специальные клавиши) для ввода желаемых команд. Можно переключаться между редакторами emacs и vi. При этом все основные команды, связанные с редактированием, остаются присвоены одним и тем же клавишам в обоих редакторах. Обычно по умолчанию устанавливается редактор emacs. Все сочетания клавиш для редактирования, описанные в этой главе, предназначены именно для работы с emacs.

Расширения названий команд и файлов

При автоматическом задании расширений для названий команд и файлов bash помогает значительно уменьшить долю рутинной печатной работы. Нужно всего лишь указать начальные буквы названия команды или файла и нажать клавишу Tab. Если название команды или файла уже можно однозначно идентифицировать, это название дописывается полностью. Если имеется несколько названий, начинающихся одинаково, то название будет дополнено не до конца. Система выдаст в таком случае звуковой сигнал, означающий, что, возможно, имя записано не до конца.

Расширение имени файла лучше всего показать на примере. Ввод

```
user$ em Tab com Tab
```

автоматически дополняется (на моем компьютере) до

```
user$ emacs command.tex
```

Здесь `emacs` — это название моего любимого редактора, а `command.tex` — имя файла LATEX с одной из глав книги, которую вы сейчас читаете. Чтобы дополнить `em`, bash производит поиск исполняемых программ по всем каталогам, указанным в переменной `PATH`. Напротив, при дополнении имени файла учитывается только текущий каталог.

Подобное расширение работает и с именем файла, перед которым указано несколько каталогов. Если задать

```
user$ ls/usr/sh Tab
```

то bash приведет данный ввод в форму

```
user$ ls/usr/share/
```

Если однозначно дополнить имя файла невозможно (система выдает звуковой сигнал), то можно просто еще раз нажать клавишу табуляции. Тогда bash перечислит под строкой для ввода все возможные варианты (в столбец). Ввод

```
user$ e Tab Tab
```

приводит к выводу практически бесконечного списка программ и команд, начинающихся на букву `e`. После этого ввод можно продолжить.

СОВЕТ

Программы и команды из текущего каталога при вводе команд учитываются лишь при условии, что этот каталог указан в переменной PATH (PATH задается в echo $PATH; текущий каталог сокращается с Помощью «.»).

В большинстве дистрибутивов Linux из соображений безопасности текущий каталог в PATH не указывается. Чтобы можно было выполнять программы из этого каталога, нужно ввести ./name.

Определение пути к программе. При автоматическом дополнении имени команды система скрывает, где именно находится программа. Узнать ее местонахождение можно несколькими способами.

- Команда `whereis` *имя* просматривает все стандартные каталоги.
- Команда `which` *имя* просматривает все каталоги, содержащиеся в переменной `PATH`, и определяет программу, которая выполнялась бы после ввода команды без указания пути. Команда `which` полезна в том случае, когда в системе есть несколько версий одной программы и эти версии расположены в разных каталогах.
- Команда `type` *имя* работает подобно `which`, но учитывает и те команды, которые интегрированы в bash или заданы в форме псевдонимов (см. подраздел «Сокращения, связанные с псевдонимами» этого раздела).

В bash предусмотрены аналогичные механизмы расширения и для имен из домашнего каталога (так, на моем компьютере `~ko` и нажатие Tab возвращает `~kofler/`) и для названий переменных (`$PAT` и нажатие Tab возвращает `$PATH`).

Программно-специфичные расширения. При выполнении команды `latex` *имя*`.tex` из всех возможных файлов рассматриваются лишь те, что имеют расширение `*.tex`. Если выполнить `man` *имя*, будут рассмотрены записи, по которым имеются тексты в справке `man`. Аналогично существуют многочисленные другие программы и команды, при работе с которыми выбор возможных файлов или параметров заранее ограничен. Разумеется, это практично, так как в таком случае при выборе расширения учитываются лишь те параметры и файлы, которые подходят данной команде.

Именно за такие соответствия отвечает команда bash `complete`. Во многих дистрибутивах предусмотрена подробная конфигурация `complete`, которая, однако, частично требует дополнительной установки (например, в Fedora существует пакет `bash-completion`). Как правило, конфигурация осуществляется в одном из следующих файлов:

- `/etc/bash_completion`;
- `/etc/bash_completion.d/*`;
- `/etc/profile.d/complete.bash`;
- `/etc/profile.d/bash_completion.sh`.

ПРИМЕЧАНИЕ

Чтобы задать собственные правила дополнения имен, вам потребуется изучить изрядно запутанный синтаксис complete. Сжатое описание синтаксиса дается в файлах help complete и man bash (выполните поиск по словам Programmable Completion).

Другие советы по конфигурированию механизма дополнения имен предлагаются на следующих сайтах: http://www.caliban.org/bash/index.shtml; http://www.pl-berichte.de/t_system/bash-completion.html.

Важные сочетания клавиш

Таблица 6.1 построена, исходя из предположения, что bash конфигурирована для работы с emacs. Именно так и обстоит дело почти во всех дистрибутивах. Если система реагирует на нажатие некоторых клавиш иначе, почитайте раздел 6.2 о конфигурировании. Если вы работаете в X, то и в том случае, когда bash конфигурирована правильно, управление клавиатурой в X может вызывать проблемы. Если вы работаете в Gnome, выполните в окне терминала Правка ▸ Комбинации клавиш и снимите флажок Деактивировать все буквенные сокращения в меню.

Таблица 6.1. Сочетания клавиш в bash

Клавиши	Значение
↓, ↑	Прокрутить последние введенные команды
←, →	Передвинуть курсор назад или вперед
Home, End	Переместить курсор в начало или в конец строки
Ctrl+A, Ctrl+F	Переместить курсор в начало или в конец строки (используется, если Pos1 и End не работают)
Alt+B, Alt+F	Переместить курсор на одно слово вперед или назад
Backspace, Delete	Удалить символы вперед или назад
Alt+D	Удалить слово

Клавиши	Значение
Ctrl+K	Удалить текст до конца строки
Ctrl+Y	Заново вставить последний удаленный текст
Ctrl+T	Поменять местами две предыдущих строки
Alt+T	Поменять местами два предыдущих слова
Tab	Дополнить названия команд и файлов
Ctrl+I	Погасить экран
Ctrl+R	Найти ранее введенные команды
Alt+«.»	Вставить последний примененный параметр
Ctrl+_	Отменить последнее изменение (Undo)

Вставка последнего параметра. Функцию сочетания клавиш Alt+. можно понять только на примере. Предположим, вы только что скопировали файл (`cp` *`имя1 имя2`*). Теперь хотите удалить сделанную копию следующей командой. Вместо `rm` *`имя2`* введите `rm`, а потом нажмите Alt+«.». Оболочка bash автоматически вставит последний примененный параметр команды. Если несколько раз нажать Alt+«.», то можно получить доступ и к некоторым последним параметрам (то есть при двукратном нажатии получаем *`имя1`*).

Поиск команд. Отдельного объяснения заслуживает сочетание клавиш Ctrl+R. Чтобы можно было искать введенные ранее команды, нажмите в начале строки Ctrl+R, а потом укажите первые символы из искомой командной строки. Если несколько раз нажать Ctrl+R, можно просмотреть различные варианты команд, подходящих под заданный шаблон. Сочетание Ctrl+S работает аналогично Ctrl+R, однако просматривает список введенных команд в обратном направлении. Клавиши ввода, табуляции и управления курсором прекращают поиск и выполняют найденную команду либо позволяют отредактировать найденную строку.

Некоторые консоли воспринимают Ctrl+S как команду, которая временно останавливает вывод. Вывод возобновляется командой Ctrl+O. Если ваша консоль так реагирует на Ctrl+S, поиск команд можно производить лишь с помощью Ctrl+R.

Все сокращения bash собраны в библиотеке `readline`, используемой оболочкой для обработки ввода. В `man readline` перечислено еще больше сокращений.

Сокращения, связанные с псевдонимами

При вводе команд для оболочки вы можете сократить себе работу по набору с помощью команды `alias`. Она определяет сокращения. При обработке командной строки система проверяет, содержится ли в первом слове сокращение. Если это так, то сокращение заменяется полным текстом.

Сокращения для определенных сочетаний параметров или имен файлов неприменимы, так как bash не ищет сокращений во всех параметрах команды. Однако bash распознает особые случаи, когда в одной командной строке перечисляется несколько команд (например, при использовании программных каналов, подстановке команд, последовательного выполнения команд с помощью `;`), и просматривает все имеющиеся названия команд — нет ли в них сокращений.

```
user$ alias cdb='cd~kofler/linuxbuch'
```

Вышеуказанная команда определяет сокращение cdb, с помощью которой я перехожу в один из самых нужных каталогов своего компьютера — ~kofler/linuxbuch.

Вызовы команды alias можно применять и в виде вложений. Обратите внимание на то, что сокращения alias имеют приоритет над одноименными командами. Это свойство можно использовать для того, чтобы воспрепятствовать нежелательному вызову команды:

```
user$ alias more=less
```

Теперь при любой попытке выполнить команду more запускается мощная программа less. Если же по каким-то причинам вам снова понадобится команда more, потребуется указать путь к ней полностью (/bin/more) или поставить перед ней обратный слэш (\more). В таком случае он препятствует интерпретации псевдонима.

Сокращения alias можно вновь удалять с помощью unalias. В противном случае сокращения остаются действительны до того, как вы покинете оболочку (то есть не позднее, чем до выхода из системы). Если после этого вам все еще будут нужны какие-либо сокращения, задайте команды alias в файлах /etc/bashrc и ~/.bashrc в вашем домашнем каталоге.

Во многих дистрибутивах некоторые сокращения псевдонимов задаются по умолчанию. Например, если все время поступает запрос от rm относительно того, следует ли удалить файл, это обычно связано с заданным псевдонимом rm=rm -i. Список всех псевдонимов, действительных в настоящий момент, возвращает команда alias. В следующих строках показано, в каких разделах дистрибутивов Debian, Fedora, SUSE и Ubuntu располагаются определения псевдонимов.

```
Debian Fedora, Ubuntu:/etc/bashrc        /etc/profile.d/*.sh   ~/.bashrc
SUSE:                 /etc/bash.bashrc   /etc/profile.d/*.sh   ~/.bashrc ~/.alias
```

Подобно сокращениям могут работать и программы оболочки. Shell-сценарии имеют и определенное достоинство — они способны разбираться в параметрах ($1, $2 и т. д.) и применяются более гибко.

6.4. Переадресация ввода и вывода

При выполнении команд в bash существуют так называемые стандартные файлы. При этом термин «файл» имеет несколько иное значение, чем обычно, — речь идет не о настоящих файлах, а о дескрипторах файлов, которые обрабатываются на уровне операционной системы как обычные файлы.

- Стандартный ввод — программа, исполняемая в настоящий момент (например, bash или любая запущенная из нее команда), считывает весь стандартный ввод. Источником стандартного ввода обычно является клавиатура.
- Стандартный вывод — сюда переадресовывается весь программный вывод (соответствующий списку файлов, выводимому командой ls). Стандартный вывод обычно отображается в окне терминала.
- Стандартные ошибки — в действующем терминале обычно показываются и сообщения об ошибках.

Все это, казалось бы, само собой разумеется — откуда, как не с клавиатуры, должен поступать ввод и где, как не в окне терминала, должны отображаться результаты выполнения программ и сообщения об ошибках? Однако обратите внимание, что ввод и вывод можно переадресовывать.

Например, возможен случай, в котором содержание текущего каталога должно не отображаться на экране, а сохраняться в файле. Таким образом, стандартный вывод должен переадресовываться в настоящий файл. В bash это делается с помощью символа >:

```
user$ ls *.tex > content
```

Сейчас в файле `content` находится список всех TEX-файлов, расположенных в текущем каталоге. Чаще всего применяется именно такой способ переадресации вывода. Однако есть еще два варианта: `2>` *файл* переадресовывает сообщения об ошибках в указанный файл; `>&` *файл* или `&>` *файл* переадресовывают в указанный файл и сообщения об ошибках, и программный вывод. Если использовать вместо > двойной символ >>, то весь ввод дописывается в конце уже имеющегося файла.

Переадресация ввода осуществляется с помощью < *файл*: команды, ожидающие ввода с клавиатуры, считывают ввод из указанного таким образом файла.

ВНИМАНИЕ

Невозможно одновременно обрабатывать файл и записывать в него же результаты обработки!

Команда `sort dat > dat` или `sort < dat > dat` приводит к удалению файла `dat`!

В табл. 6.2 показаны возможности переадресации ввода и вывода.

Таблица 6.2. Переадресация ввода и вывода

Команда	Функция
Команда > файл	Переводит стандартный вывод в указанный файл
Команда < файл	Считывает ввод из указанного файла
Команда 2> файл	Переводит сообщения об ошибках в указанный файл
Команда >& файл	Переадресовывает вывод *и* ошибки
Команда &> файл	Переадресовывает вывод *и* ошибки
Команда >> файл	Дописывает стандартный вывод в конце имеющегося файла
Команда &>> файл	Дописывает стандартный вывод и ошибки в конце имеющегося файла (начиная с версии bash 4.0)
Команда1 | Команда2	Передает вывод команды1 команде2
Команда | tee файл	Показывает вывод и одновременно сохраняет копию файла

Программные каналы

Программный канал (pipe) создается с помощью символа |. При этом вывод первой команды используется как ввод для второй команды. На практике программные каналы часто объединяются командой `less`, если нужно просматривать длинный ввод в виде постраничной разбивки.

```
user$ ls -l | less
```

Приведенная команда позволяет получить «оглавление» текущего каталога и записать его в программный канал. Оттуда команда less, выполняемая параллельно, считывает ввод предыдущей команды и отображает его на экране.

Программные каналы великолепно подходят для комбинирования различных команд. Так, например, следующая команда возвращает отсортированный список всех установленных пакетов RPM:

```
user$ rpm -qa | sort
```

Вместо программных каналов для переадресации ввода и вывода могут использоваться так называемые FIFO-файлы. FIFO означает «первым пришел — первым обслужен» (first in first out) и реализует идею программного канала в форме файла. Работать с FIFO-файлами при вводе значительно сложнее, чем с программными каналами, однако они позволяют уяснить, на что же именно влияет символ |. На практике такие файлы применяются для того, чтобы две программы, работающие независимо друг от друга, могли обмениваться информацией.

```
user$ mkfifo fifo
user$ ls -l > fifo &
user$ less < fifo
```

С помощью трех вышеуказанных команд сначала создается FIFO-файл. Кроме того, в виде фонового процесса запускается ls. Она записывает свой вывод в файл. Оттуда less вновь считывает данные и отображает их на экране.

Только такие команды подходят для формулирования программного канала, из которого команды, предназначенные для ввода, будут считывать информацию. Если вы имеете дело с иной ситуацией, можно достичь сходных эффектов с помощью подстановки команд или команды xargs.

Размножение вывода командой tee

Иногда программный вывод нужно сохранить в файле, однако одновременно с этим выполнение программы необходимо отслеживать на экране. В таком случае требуется удвоение ввода, причем одна копия отображается на экране, а вторая сохраняется в файле. Эту задачу выполняет команда tee:

```
user$ ls | tee content
```

Содержание текущего каталога отображается на экране и одновременно сохраняется в файле content. При этом сначала происходит переадресация стандартного вывода к команде tee. По умолчанию стандартный вывод отображается на экране, а копия этой информации сохраняется в указанном файле. Поскольку в данном случае речь в действительности идет о размножении вывода, обратите внимание, как переадресовать стандартный вывод tee в файл:

```
user$ ls | tee content1 > content2
```

В итоге получаются два одинаковых файла: content1 и content2. Предыдущая команда приведена просто для примера. Следующий пример понять сложнее, но он несколько ближе к практике:

```
user$ ls -l | tee content1 | sort +4 > content2
```

В content1 расположено обычное оглавление, которое автоматически отсортировано командой ls по имени. Копия этого вывода передается sort, а затем происходит сортировка по размеру файла (пятый столбец, то есть параметр +4) и сохранение информации в файле content2.

6.5. Выполнение команд

Обычно для запуска команды необходимо ввести ее имя. В командной строке можно указать сколько угодно специальных символов, которые будут интерпретированы bash еще до запуска команды. Таким же образом можно запускать команды в фоновом режиме, охватывать с помощью подстановочных символов (джокеров) одновременно много похожих имен файлов (например *.tex), подставлять результаты выполнения одной команды в список параметров другой команды и т. д.

Фоновые процессы. Важнейший и самый востребованный подстановочный символ — &. Если указать его в конце командной строки, bash запустит программу в фоновом режиме. Это имеет смысл в первую очередь тогда, когда на выполнение команды уходит много времени, чтобы можно было продолжать работу, не дожидаясь окончания выполнения такой программы.

```
User$ find / -name '*sh' > result &
[1] 3345
```

Вышеуказанная команда ищет по всей файловой системе файлы, названия которых оканчиваются на sh. Список найденных файлов записывается в файле result. Поскольку команда выполняется в фоновом режиме, работу можно не прерывать. Вывод [1] 3345 означает, что фоновый процесс имеет номер PID 3345. Здесь PID означает идентификатор процесса (process ID). Номер PID интересен в том случае, когда процесс был аварийно завершен командой kill. Номер в квадратных скобках — это номер фонового процесса, запущенного в bash. Как правило, этот номер не важен.

Если при запуске команды вы забудете поставить символ &, не следует ни дожидаться окончания выполнения программы, ни принудительно завершать программу нажатием Ctrl+C. Гораздо лучше приостановить выполнение программы, нажав Ctrl+Z, а затем продолжить ее работу в виде фонового процесса с помощью команды bg.

Выполнение нескольких команд. После символа & вы также можете указать следующую команду. В таком случае первая команда будет выполняться в фоновом режиме, а вторая — на виду. В следующем примере рассмотренная выше команда find вновь запускается в фоновом режиме. Однако ls одновременно выводит содержание текущего каталога:

```
user$ find / -name '*sh' > result & ls
```

Если вместо символа & поставить точку с запятой, то bash поочередно выполнит команды в фоновом режиме:

```
user$ ls; date
```

Эта команда сначала отображает содержание текущего каталога, а затем выводит текущий файл. Если необходимо перенаправить всю эту информацию в файл с помощью >, то обе команды ставятся в круглых скобках. В таком случае они выполняются одной и той же оболочкой.

```
user$ (ls; date) > content
```

В файле content теперь находится список файлов, созданный ls, а также текущая дата, выясненная date. Благодаря круглым скобкам обе команды выполняются одной и той же оболочкой и выдают общий результат (как правило, ситуация иная — при запуске каждой новой команды активируется новая оболочка).

Используя сочетания символов && и ||, можно выполнять команды относительно, то есть в зависимости от результата другой команды (табл. 6.3).

```
user$ команда1 && команда2
```

Выполняет *команду1*. Только в случае, если эта команда была выполнена успешно (без ошибки, без выдачи 0 в качестве возвращаемого значения), выполняется команда2.

```
user$ команда1 || команда2
```

Таблица 6.3. Выполнение команд

Команда	Функция
Команда1 ; команда2	Выполняет команды одну за другой
Команда1 && команда2	Выполняет команду2, если команда1 была выполнена успешно
Команда1 || команда2	Выполняет команду2, если команда1 возвращает ошибку
Команда &	Запускает команду в фоновом режиме
Команда1 & команда2	Запускает команду1 в фоновом режиме, команду2 — на виду
(Команда1 ; команда2)	Выполняет обе команды в одной и той же оболочке

Другие возможности для создания условий и ветвления команд связаны с использованием оболочковой команды if, которая, однако, будет интересна лишь тем, кто собирается заниматься программированием на языке оболочки.

6.6. Механизмы подстановки

Термин «механизм подстановки» кажется очень абстрактным и сложным. Основная идея заключается в том, что команды, образуемые с применением специальных символов, заменяются их результатами. В простейшем случае это означает: при интерпретации команды ls *.tex последовательность символов *.tex заменяется списком подходящих файлов, а именно: buch.tex, *команда*.tex.

Цель этого раздела — кратко описать важнейшие механизмы, используемые при интерпретации команд (см. обобщенную информацию в табл. 6.4), а именно: джокерные символы, применяемые для образования имен файлов, фигурные скобки для объединения последовательностей символов, квадратные скобки для вычисления выражений, содержащихся в арифметических скобках, обратные апострофы для подстановки команд и т. д.

Таблица 6.4. Механизмы подстановки

Команда	Функция
?	Любой символ
*	Любое количество любых символов (в том числе ни одного), но не *-файлы!
**	Любые файлы и каталоги, в том числе из всех подкаталогов (начиная с версии bash 4.0 — shopt -s globstar)
[abc]	Один из символов, указанных в скобках
[a–f]	Символ из указанного диапазона
[!abc]	Любые символы, кроме тех, что указаны в скобках
[^abc]	Аналогично предыдущему
~	Сокращенное обозначение домашнего каталога
.	Текущий каталог
..	Каталог на один уровень выше
ab{1,2,3}	Возврат ab1 ab2 ab3
a{1..4}	Возврат a1 a2 a3 a4
$[3*4]	Арифметические вычисления
`команда`	Замена команды результатом ее выполнения
$(команда)	Вариант, аналогичный предыдущему
Команда "символ"	Отмена интерпретацию любых специальных символов, кроме $
Команда 'символ'	Похоже на предыдущий вариант, но с большими ограничениями (не допускает подстановки переменных)

Механизм подстановки, применяемый в данном случае, называется подстановкой параметров. С помощью этого метода можно анализировать и изменять последовательности символов, сохраненные в переменных. Общий синтаксис таков: `${var__text}`, где `var` — это имя переменной, `_` — один или два специальных символа, а `text` — шаблон для поиска или установка по умолчанию. Этот механизм подстановки подробнее рассматривается в подразделе «Подстановка параметров» раздела 6.10.

Образование имен файлов с помощью * и ?. Если вы укажете `rm *.bak` и команда `rm` действительно удалит все ваши файлы, чьи названия оканчиваются на `.bak`, то это случится именно из-за bash. Оболочка просматривает текущий каталог в поисках подходящих файлов и заменяет `*.bak` соответствующими именами файлов.

В качестве джокерных символов используются `?` (что означает любой символ) и * (что означает любое количество любых символов, в том числе ни одного). Последовательность символов `[a,b,e-h]*` означает файлы, имена которых начинаются с `a`, `b`, `e`, `f`, `g` или `h`. Если первый символ, указанный в квадратных скобках, — это `^` или `!`, то допустимы все символы, кроме указанных в скобках. Символ ~ можно использовать в качестве сокращенного названия домашнего каталога.

Функции специальных символов можно протестировать с помощью следующей команды `echo`. Первая команда возвращает все файлы и каталоги, находящиеся в корневом каталоге. Вторая команда ограничивает вывод файлами и каталогами, имена которых начинаются с букв от `a` до `f`.

```
user$ echo /*
/bin /boot /dev /etc /home /lib /lost+found /media /misc /mnt /net /opt
/proc /root /sbin /selinux /srv /sys /tmp /usr /var
```

```
user$ echo /[a-f]*
/bin /boot /dev /etc
```

Поскольку за образование имен файлов отвечает не конкретная программа, а оболочка bash, результаты могут выглядеть иначе, чем вы, возможно, ожидаете. Так, `ls *` может возвратить практически бесконечный список файлов, даже если в текущем каталоге их всего несколько. Команда `ls` после дополнения с помощью * возвратит список всех файлов и каталогов. Кроме того, `ls` отображает не только имена каталогов, но и их содержимое! Если вы хотите получить простой список всех файлов и каталогов, используйте параметр `-d`. Он обеспечивает то, что содержимое каталогов, выводимых в строке параметров, отдельно отображаться не будет.

Если вы хотите обеспечить обратную связь с системой и отслеживать внутреннее функционирование bash, можете выполнить команду `set -x`. Тогда bash будет отображать выполнение всех команд по мере их интерпретации в командной строке (с любыми параметрами, которые могут предшествовать команде и с дополняемыми именами файлов).

По умолчанию * не учитывает файлов и каталогов, имена которых начинаются с точки (иными словами, скрытых). Если вам требуется такой эффект, необходимо воспользоваться bash-параметром `glob_dot`:

```
user$ shopt -s dotglob
user$ echo *
...
user$ shopt -u dotglob      (dotglob снова деактивирована)
```

Образование имен файлов с помощью **. Начиная с версии 4.0, сочетание символов ** рекурсивно охватывает файлы и каталоги. Из соображений совместимости эта новая функция по умолчанию деактивирована. Если вы хотите ею воспользоваться (например, в сценарии), следует установить параметр bash `globstar` с помощью `shopt -s`.

```
user$ shopt -s globstar
user$ echo **
...
```

Образование последовательностей символов с помощью {}. Из последовательностей символов, заключенных в фигурные скобки, bash образует любые мыслимые последовательности символов. Официальное название такого механизма подстановки — раскрытие скобок (brace expansion). Выражение `part{1,2a,2b}` означает `part1 part2a part2b`. Пользуясь раскрытием скобок, можно облегчить печатную работу, если вы обращаетесь ко многим файлам и каталогам с похожими именами. По сравнению с джокерными символами * и ? есть определенное преимущество — можно образовывать имена еще не существующих файлов (это касается, например, `mkdir`).

```
user$ echo {a,b}{1,2,3}
a1 a2 a3 b1 b2 b3

user$ echo {ab,cd}{123,456,789}-{I,II}
ab123-I ab123-II ab456-I ab456-II ab789-I ab789-II
cd123-I cd123-II cd456-I cd456-II cd789-I cd789-II
```

Перечисления можно красиво оформлять в виде {a..b}, причем a и b, на ваш выбор, могут быть как буквами, так и числами. Следующие примеры объясняют принцип работы этой функции лучше любых описаний:

```
user$ echo {1..5}
1 2 3 4 5

user$ echo {z..t}
z y x w v u t
```

Вычисление арифметических выражений с помощью []. Обычно в bash нельзя производить вычисления. Если написать 2 + 3, то оболочка «не сообразит», что делать с этим выражением. Если вы хотите вычислять, работая с оболочкой, нужно заключить выражение в квадратные скобки и поставить перед ними символ $:

```
user$ echo $[2+3]
5
```

В квадратных скобках можно использовать большинство операторов, известных из языка программирования C: +, -, *, / для четырех основных арифметических операций, % для вычислений по модулю, ==, !=, <, <=, > и >= для сравнений, << и >> для перемещений битов, !, && и || для логических операторов NO, AND и OR и т. д. Все вычисления выполняются с 32-битными целыми числами (диапазон чисел: ±2 147 483 648). Если следует извлечь отдельные значения из переменных перед переменными, необходимо ставить символ $.

Есть еще один способ производить вычисления — с помощью команды expr. Это самостоятельная команда Linux, работающая независимо от bash.

Подстановка команд. Пользуясь подстановкой команд, можно заменить команду в командной строке результатом этой команды. Для этого команда должна быть заключена между двумя символами `. Альтернативная запись — $(*команда*). Второй метод предпочтителен, так как, во-первых, в таком случае вы избегаете путаницы с тремя разными кавычками (", ' и `), во-вторых, второй метод допускает вложения.

Обозначенная таким образом команда будет заменена результатом ее выполнения. Подобная запись обеспечивает вложенный вызов нескольких команд, причем одна команда передает результат своего выполнения другой. Две представленные далее равнозначные команды проясняют принцип работы этого чрезвычайно мощного механизма:

```
user$ ls -lgo  `find /usr/share -name '*README*'`
user$ ls -lgo $(find /usr/share -name '*README*')
```

Вышеуказанная команда сначала выполняет find /usr/share -name '*README*'. Результат выполнения этой команды — список всех файлов, находящихся в каталоге /usr/share и содержащих последовательность символов README. Теперь этот список вставляется в командную строку на месте команды find. Тогда командная строка может записываться следующим образом:

```
user$ ls -lgo /usr/share/a2ps/ppd/README \
> /usr/share/a2ps/README ...
```

Эта команда приводит к следующему результату:

```
-rw-r--r-- 1  301  15. Feb 12:30  /usr/share/a2ps/ppd/README
-rw-r--r-- 1  1029 15. Feb 12:30  /usr/share/a2ps/README
...
```

Получить такой результат, используя обычный программный канал с символом |, не удастся. Команда ls не ожидает никакого стандартного ввода, а также игнорирует информацию, передаваемую find через программный канал. И именно поэтому следующая команда отображает только содержимое текущего каталога. Результаты find выводиться не будут!

```
user$ find /usr/share -name '*README*' | ls -l (не работает!)
```

Есть еще одно решение, позволяющее обойтись без подстановки команд: с помощью команды xargs данные стандартного ввода передаются указанной команде:

```
user$ find /usr/share -name '*README*' | xargs ls -l
```

Важное достоинство xargs заключается в том, что объем данных, которые можно переработать, ничем не ограничен. При необходимости xargs вызывает команду неоднократно и передает получаемые данные в стандартный ввод в несколько этапов. Напротив, при подстановке команд максимальная длина командной строки ограничена, как правило, несколькими тысячами символов.

Если в именах файлов содержатся пробелы, передача таких имен может вызывать затруднения. Чтобы избежать этих проблем, передайте команде find параметр -print0, а команде xargs — параметр -null. Следующая команда ставит для всех каталогов бит execute:

```
user$ find -type d -print0 | xargs --null chmod a+x
```

Специальные символы в последовательностях. Поскольку в bash практически любой символ, за исключением букв и цифр, имеет специальное значение, представляется практически невозможным использовать такие знаки в последовательностях символов и именах файлов. Эта проблема решается двумя способами. Можно поставить перед специальным символом обратный слэш (\) либо заключить всю последовательность символов в апострофы или кавычки. Указав апострофы, вы можете, например, удалить файл с именем ab* $cd:

```
user$ rm 'ab* $cd'
```

Обратите внимание: символ ' используется для обозначения последовательностей символов, а ` — для подстановки команд (см. выше). Эти символы не равнозначны!

Кавычки работают почти так же, как и апострофы. В любом случае они накладывают меньше ограничений и позволяют использовать некоторые специальные символы ($, \ и `). Поскольку последовательность символов стоит в кавычках, интерпретируются оболочковые переменные, перед которыми стоит символ $:

```
user$ echo "This is the access path: $PATH"
```

Команда возвращает в качестве результата последовательность символов This is the access path, за которой следует содержимое оболочковой переменной PATH.

Если применить не кавычки, а обычные апострофы, то вся последовательность символов будет передана через echo в неизменном виде.

6.7. Оболочковые переменные

Функциональность bash и многих других программ Linux управляется состояниями так называемых переменных оболочки. Такие переменные можно сравнить с переменными языка программирования, однако в них можно сохранять только последовательности символов. Присваивание переменных оболочки осуществляется с помощью оператора присваивания =. Чтобы отобразить содержимое оболочковой переменной, нужно воспользоваться командой echo, при этом перед именем переменной необходимо поставить символ $:

```
user$ var=abc
user$ echo $var
abc
```

При присваивании переменных нельзя оставлять между оператором присваивания = и именем переменной пробелов. Запись var = abc синтаксически неверная и работать не будет!

Если в содержимом оболочковой переменной должны содержаться пробелы или другие специальные символы, то при присваивании всю последовательность символов необходимо заключить в одиночные кавычки:

```
user$ var='abc efg'
```

При присваивании можно записывать друг за другом сразу по несколько последовательностей символов. В следующем примере переменной a присваивается новая последовательность символов, состоящая из содержимого этой переменной, последовательности символов xxx и еще раз из исходного содержимого:

```
user$ a=3
user$ a=$a'xxx'$a user$ echo $a 3xxx3
```

В следующем примере имеющаяся переменная PATH (со списком всех каталогов, в которых может осуществляться поиск исполняемых программ) дополняется в домашнем каталоге каталогом bin. Теперь вы можете выполнять любые команды, находящиеся в этом каталоге (не указывая путь полностью).

```
user$ echo $PATH
/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin
user$ PATH=$PATH':/home/kofler/bin'
user$ echo $PATH
/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin:/home/kofler/bin
```

Вычисления с переменными можно производить в квадратных скобках описанным выше способом:

```
user$ a=3
user$ a=$[$a*4]
user$ echo $a 12
```

Если результат выполнения команды следует сохранить в переменной, нужно произвести описанную выше подстановку команды по образцу `$(команда)`. В следующем примере текущий каталог сохраняется в `a`:

```
user$ a=$(pwd)
user$ echo $a /home/kofler
```

Содержимое переменных сохраняется только в оболочке. При выходе из оболочки эта информация теряется. Если определенные переменные требуются вам снова и снова, то присваивание следует производить в файле `/etc/profile` или `.profile` домашнего каталога. Оба этих файла (если они имеются) автоматически выполняются при запуске bash.

Если вы хотите сохранить содержимое одной из переменных в файле, то нужно выполнить команду `echo` с переадресацией ввода:

```
user$ echo $var > file
```

Локальные и глобальные переменные (переменные окружения)

Термины «локальный» и «глобальный» для описания переменных взяты из мира языков программирования. Переменная оболочки считается глобальной, если она передается далее при запуске команды или программы оболочки. Часто глобальные переменные именуются *переменными окружения* (environment variables).

Обратите внимание, что все переменные, полученные в результате обычного присваивания, могут быть только локальными! Чтобы задать глобальную переменную, следует выполнить `export` или `declare -x`.

Для управления переменными в оболочке существуют многочисленные команды, которые функционально иногда дублируют друг друга. Например, для объявления глобальной переменной можно использовать как `export`, так и `declare -x`. Следующие примеры приводятся для того, чтобы немного прояснить ситуацию с дублирующимися командами:

```
a=3             Краткий способ записи let, переменная a является локальной
declare a=3     Присваивает локальной переменной a значение (как let)
declare -x a=3  Присваивает локальной переменной a значение (как export)
export          Отображает все глобальные переменные
export a        Делает переменную a глобальной
export a=3      Присваивает глобальной переменной a значение
let a=3         Присваивает глобальной переменной a значение
local a=3       Определяет переменную a как локальную (лишь в функциях оболочки)
printenv        Как и export, отображает все глобальные переменные
set             Отображает все переменные (и локальные, и глобальные)
unset a         Удаляет переменную a
```

Если вы создаете переменные, которые должны управлять работой других команд Linux, эти переменные всегда должны быть глобальными! Чтобы можно было, с одной стороны, использовать механизмы подстановки, действующие в оболочке, а с другой — задавать глобальные переменные, эти переменные следует сначала

присваивать с помощью x=..., а потом дополнительно определять их как глобальные с помощью export x.

Присваивание переменных всегда остается действительным *только для одной* оболочки. Если вы работаете с несколькими терминалами или окнами терминалов, в каждом из них функционирует самостоятельная оболочка, не зависящая от других. Изменение переменной в оболочке никак не влияет на другие оболочки. Однако часто используемые присваивания переменных можно записать в файле ~/.profile, который будет автоматически выполняться при запуске любой оболочки.

Важнейшие оболочковые переменные

В принципе можно ввести сколько угодно новых переменных, назвать их на свой вкус и использовать по собственному разумению. При этом не следует применять переменные, которые уже существуют, так как они обычно интерпретируются bash, а зачастую и другими командами Linux. При неконтролируемом изменении таких переменных можно повредить механизм обработки команд — Linux вдруг разучится находить файлы и т. д.

В этом разделе в алфавитном порядке будут описаны важнейшие оболочковые переменные.

- BASH — содержит имя файла bash.
- HOME — содержит путь к домашнему каталогу, например /home/mk.
- LOGNAME — хранит логин (имя пользователя).
- HOSTNAME — содержит хост-имя (имя компьютера).
- MAIL — включает путь к каталогу, в котором сохраняется входящая почта (только если у вас установлен локальный почтовый сервер).
- OLDPWD — содержит путь к последним использовавшимся каталогам.
- PATH — хранит список каталогов. Если bash должна выполнить команду, она просматривает все каталоги, перечисленные в PATH, в поисках команды. Каталоги отделяются друг от друга двоеточиями.

  Настройка PATH отличается от дистрибутива к дистрибутиву, в различных местах системы в ходе запуска (Init-V, Upstart). Лучше всего производить собственные изменения в /etc/profile или (если в вашем дистрибутиве предусмотрена такая возможность) в файле каталога /etc/profile.d. Туда необходимо вставить команду в соответствии со следующим образцом:

  ```
  # Дополнение в /etc/profile или в /etc/profile.d/myown.sh
  PATH=$PATH:/myown/bin
  ```

  Из соображений безопасности (чтобы избежать незапланированного выполнения программ в текущем каталоге), в PATH не указывается локальный каталог. Если вы хотите выполнять программы, содержащиеся в текущем каталоге, не указывая перед ними ./, нужно дополнить PATH точкой.
- PROMPT_COMMAND — может содержать команду, выполняемую всякий раз, когда bash отображает приглашение командной строки.

- PS1 — хранит последовательность символов, содержимое которой отображается рядом с началом каждой строки ввода (эта последовательность называется подсказкой, или приглашением). В ней предусмотрены в числе прочих следующие последовательности символов: \t — текущее время, \d — сегодняшняя дата, \w — текущий каталог, \W — последняя часть текущего каталога (например, X11 для /usr/bin/X11), \u — имя пользователя, \h — хост-имя (имя компьютера), а также символ подсказки \$ (\$ — для обычных пользователей, # — для администратора).
- PS2 — похожа на PS1, но эта последовательность символов обычно отображается только для многострочного ввода (то есть если первая строка завершается символом \). Типичная настройка — ">".
- PWD — содержит путь к текущему каталогу.

Кроме описанных здесь, обычно используются и многие другие переменные окружения, которые управляют функциями оболочки, а также многих других программ. Список всех заданных переменных выводится с помощью команды printenv | sort.

6.8. Примеры сценариев bash

Программы оболочки — это обычные текстовые файлы, в которых записано по несколько команд Linux или bash. После запуска программы оболочки эти команды выполняются по очереди. Программе оболочки можно передавать параметры как обычной команде. Эти параметры могут интерпретироваться внутри программы.

Поскольку при обычном последовательном выполнении нескольких команд сужается поле для маневра и решения сложных задач, bash поддерживает программирование оболочки, используя команды, создающие условные переходы и циклы. Это уже элементы настоящего языка программирования, но для работы с ним вам не требуется ни компилятор, ни знание языка C. (Следует отметить, что это сравнение не совсем справедливо: программы на C выполняются несравнимо быстрее, поддерживают много типов переменных, умеют работать со многими специальными функциями и т. д. И все же возможностей bash предостаточно для решения поразительного множества вопросов.)

Программы оболочки обычно применяются для автоматизации востребованных последовательностей команд, которые используются при установке программ, администрировании системы, резервном копировании, конфигурации и выполнении отдельных программ и т. д.

На следующих страницах я сделаю лишь краткое введение в bash-программирование. Подробная дополнительная информация и многочисленные примеры предлагаются на отличном сайте http://bash-hackers.org/.

Dash. Чтобы повысить скорость выполнения сценариев, в Ubuntu вместо bash по умолчанию используется оболочка dash:

```
> ls -l /bin/sh
lrwxrwxrwx 1 root root ... /bin/sh -> dash
```

Конечно, в некоторых случаях dash эффективнее, чем bash, но она не полностью совместима с системой. Если вы хотите выполнить сценарий с помощью bash, то в первой его строке нужно специально указать /bin/bash вместо /bin/sh:

```
#!/bin/bash
```

Поиск скриптов. Linux просто наполнена примерами использования bash, хотя, возможно, раньше вы этого и не замечали. Многие команды, которые вы выполняли при установке, конфигурировании и администрировании Linux, на самом деле были программами bash.

Следующая команда find/grep просматривает каталог /etc/ в поисках программ оболочки. При этом опознаются все файлы, помеченные как исполняемые и содержащие последовательность символов \#! ... sh. Список всех файлов сохраняется в файле list. На выполнение команд требуется некоторое время, так как поиск производится по всей файловой системе.

```
user$ find /etc -type f -perm +111 -exec grep -q '#!.*sh' {} \; -print > shellscripts
```

Пример 1: grepall

Допустим, вы часто пользуетесь командами grep и find, чтобы искать в текущем каталоге и во всех подкаталогах файлы, содержащие определенную последовательность символов. Правильно оформленная команда выглядит следующим образом:

```
user$ find . -type f -exec grep -q searchtext {} \; -print
```

Если вам всякий раз приходится заново угадывать, какая комбинация параметров для этого необходима, лучше задать новую команду grepall, которая будет заниматься выполнением как раз этой задачи. Для этого запустите свой любимый текстовый редактор и создайте в нем текстовый файл grepall. Файл будет содержать всего две строки:

```
#!/bin/sh
find . -type f -exec grep -q $1 {} \; -print
```

Особенно важна первая строка, начинающаяся с #!. Она указывает программное имя интерпретатора, который должен выполнять сценарный файл.

СОВЕТ

Если вы не хотите лишний раз вызывать редактор, можно создать файл и с помощью cat: введите команду cat > grepall. Теперь команда ожидает стандартный ввод (данные с клавиатуры) и записывает их в файл grepall. Укажите команду вместе со всеми ее параметрами. Кроме того, завершите работу cat нажатием Ctrl+D (это соответствует EOF, то есть концу файла (end of file)). Полученный в итоге файл можно просмотреть с помощью команды cat grepall.

При попытке выполнить созданный файл grepall вы получите сообщение об ошибке permission denied. Причина этого в том, что в новых файлах обычно деактивированы биты доступа, предназначенные для выполнения файла (x). Однако эту настройку можно быстро отменить с помощью команды cmod. Теперь grepall abc возвратит желаемый список всех файлов, в которых содержится последовательность символов abc.

```
user$ ./grepall abc
bash: ./grepall: Permission denied
user$ chmod a+x grepall
user$ ./grepall abc
./bashprg.tex
```

Чтобы можно было выполнять команду grepall независимо от текущего каталога (то есть вводить просто grepall и не указывать предшествующий каталог), нужно скопировать команду в один из каталогов, содержащийся в переменной $PATH. Если эта команда должна быть доступна для всех пользователей, воспользуйтесь путем /usr/local/bin:

```
root# cp grepall /usr/local/bin
```

Пример 2: stripcomments

Второй пример также состоит из одной строки. Мы передаем команде stripcomments текстовый файл. Теперь три вложенные команды grep удаляют все строки, которые начинаются с символов # либо ; или вообще пусты. Комментарии удаляются и в том случае, когда перед ними находятся символы # или ;, а также символы пробела и табуляции. Эта команда отлично подходит для того, чтобы удалять в конфигурационных файлах все строки комментариев и отображать только настройки, актуальные в данном конкретном случае.

```
#!/bin/sh
grep -v ^[[:space:]]*\# $1 | grep -v ^[[:space:]]*\; | grep -v ^$
```

Коротко поясню: шаблон ^[[:space:]]*\# находит строки, начинающиеся с #, причем между началом строки (^) и символом # может находиться сколько угодно знаков пробела и табуляции. Аналогичным образом, вторая команда grep удаляет все строки, начинающиеся с ;. Наконец, третья команда grep удаляет все строки, состоящие только из символа начала строки и конца строки ($).

Параметр -v инвертирует обычную функцию grep: теперь, вместо извлечения найденных строк, grep выдает все строки, *не* соответствующие шаблону. Параметр -E активирует расширенный синтаксис grep, при котором можно комбинировать несколько поисковых результатов символом |.

Пример 3: applysedfile

Два предыдущих примера позволяют понять, как можно облегчить себе работу по набору текста, а также умственную работу, однако нисколько не открывают широких возможностей сценарного программирования. Следующий пример уже гораздо лучше позволяет разобраться в этом вопросе. Предположим, у вас есть целый набор файлов и в каждом файле нужно провести несколько одинаковых процессов поиска и замены. Обычно это случается, когда требуется изменить названия переменных и процедур в программном коде, который содержится во многих файлах.

При решении таких задач вам поможет сценарная программа applysedfile. Она вызывается следующим образом:

```
user$ applysedfile *.tex
```

Теперь программа создаст из всех файлов *.tex резервные копии *.bak. Дополнительно используется команда UNIX sed, чтобы выполнить для каждого файла *.tex целую последовательность команд. Эти команды должны находиться в файле ./sedfile, который автоматически используется командой applysedfile. Код applysedfile выглядит так:

```
#! /bin/bash
# Пример sedfile
# Использование: applysedfile *.tex
# Применяется ./sedfile с перечнем передаваемых файлов
for i in $*
do
 echo "process $i"
 # сделать резервную копию старого файла
 cp $i ${i%.*}.bak
 # создать новый файл
 sed -f ./sedfile < ${i%.*}.bak > $i done
```

Несколько коротких замечаний о том, как работает эта программа: четыре первых строки — это комментарии, вводимые символом #.

Ключевое слово for вводит цикл. При каждом выполнении цикла значением переменной i становится имя файла. Полный список имен файлов взят из $*. Данная комбинация является подстановочным символом для всех параметров и имен файлов, сообщаемых программе.

В теле цикла выводится имя каждого файла. С помощью cp создается резервная копия файла. При этом удаляются все символы, идущие после первой точки в имени файла, и имя файла дополняется расширением .bak. Наконец, для файла выполняется команда sed, при этом используется управляющий файл sedfile с локального компьютера.

Например, при замене буквосочетаний в немецкоязычном файле, чтобы текст соответствовал новым орфографическим правилам, первые строки рассмотренного файла будут выглядеть так:

```
s.da..dass.g s.mu..muss.g s.pa.t.passt.g s.la.t.lasst.g
```

В каждой строке мы видим команду sed, с помощью которой первая последовательность символов заменяется второй (команда s). Последующая буква g означает, что данная команда должна выполняться в одной строке многократно (например, если в строке несколько раз встретятся слова daß[1] или muß[2]).

Пример 4: сценарий резервного копирования

Следующий сценарий автоматически выполняется каждую ночь на моем root-сервере. Сначала инициализируется переменная m, в которой в виде числа указан текущий месяц. (Команда date возвращает актуальную дату и текущее время. Форматирующая строка +%m при этом извлекает месяц.)

---

1 Что (нем.).

2 Должен (нем.).

Теперь tar создает резервную копию каталога /var/www. Архив не сохраняется в конкретном файле, а с помощью | передается команде curl. Команда curl переносит данные на FTP-сервер (имя пользователя — kofler, пароль — xxxx, IP-адрес — 1.2.3.4). На FTP-сервере резервная копия сохраняется под именем www-month.tgz.

Таким образом, на протяжении года я создаю ежемесячные резервные копии. Поэтому при необходимости я могу восстановить какое-нибудь старое состояние моего сайта. Кроме того, файлы резервных копий занимают сравнительно мало места (в каждый момент времени у меня сохранено не более 12 таких копий, то есть с www-01.tgz по www-12.tgz).

Команда mysqldump создает резервную копию моей базы данных MySQL, которая называется cms. Здесь, в системе управления содержимым (CMS) моего сайта, сохраняются все его страницы и масса других данных. Опять же резервная копия передается с помощью | команде curl и сохраняется на моем FTP-сервере.

```
#!/bin/sh
m=$(date "+%m")
cd /var
tar czf - www | curl -T - -u kofler:xxxx ftp://1.2.3.4/www-$m.tgz
mysqldump -u cms -pxxxx cms | curl -T - -u kofler:xxxx ftp://1.2.3.4/cms-$m.sq
```

Весь сценарий я сохраняю в файле под именем /etc/myscripts/backup. За ежедневный вызов сценария отвечает программа Cron (подробнее см. в разделе 8.6). Соответствующий конфигурационный файл /etc/cron.d/backup выглядит так:

```
# Каждое воскресенье в 3:15
153 * * 0 root /etc/myscripts/backup
```

Пример 5: создание эскизов

Эскизами (thumbnails) называются миниатюрные версии файлов изображений. Следующий сценарий вызывается в форме makethumbs *.jpg. Он создает подкаталог 400 × 400 и сохраняет там уменьшенные копии исходных изображений. Максимальный размер новых изображений составляет 400 × 400 пикселов, причем пропорции оригинала остаются неизменными. Более мелкие изображения не увеличиваются.

Сценарий использует команду convert из пакета Image Magick. Для уменьшения изображения применяется параметр -resize. Параметр -size указывается только для ускорения обработки.

```
#!/bin/sh
# Использование: makethumbs *.jpg
if [ ! -d 400x400 ]; then# Создание подкаталога
mkdir 400x400
fi
for filename do          # Обработка всех файлов
echo "processing $filename"
convert -size 400x400 -resize 400x400 $filename 400x400/$filename
done
```

6.9. Синтаксис сценариев bash

Все файлы оболочки должны начинаться со строки, состоящей из символов #! и имени желаемой оболочки. В таком случае для выполнения файла автоматически вызывается указанная оболочка. Для большинства сценариев оболочки выбирается #!/bin/sh. Только если вы пользуетесь bash-специфичными функциями, нужно специально указать #!/bin/bash.

Сценарии оболочки могут выполняться лишь тогда, когда в них установлены биты доступа, обеспечивающие чтение (r) и исполнение (x). Если сценарии расположены на внешних носителях или сегментах дисков, нужно убедиться, что к дереву каталогов подключен параметр exec.

ВНИМАНИЕ

В первой строке сценария нельзя использовать специфические специальные символы национальных алфавитов, даже в комментариях. Оболочка bash отказывается исполнять файл и выводит сообщение cannot execute binary file (не удается выполнить двоичный файл).

В файлах сценариев нельзя разделять строки типичными для Windows сочетаниями команд возврата каретки (carriage return) и перехода на новую строку (linefeed). Если файл был создан в Windows и скопирован в Linux, такие сочетания могут иногда попадаться. В таком случае bash вернет немногим более понятное сообщение об ошибке bad interpreter (плохой интерпретатор).

При работе с файлами Unicode (UTF8) правильное разбиение на строки обеспечивается следующей командой:

```
recode u8/cr-lf..u8 < windowsfile > \ linuxfile
```

Если вы пишете коллекцию собственных сценарных программ для ежедневного использования, целесообразно сохранять эти сценарии в одном и том же месте. В качестве каталога подойдет ~/bin.

Если дополнительно внести такие же изменения в файл .profile, эти сценарные программы могут выполняться и без указания полного пути.

В некоторых дистрибутивах делать это не нужно, в них ~/bin всегда входит в состав PATH.

```
# Addition in ~/.profile or in ~/.bashrc
PATH=$PATH':~/bin'
```

6.10. Переменные в сценариях bash

Вводная информация по работе с переменными уже давалась в разделе 6.7. Я объяснил, в частности, разницу между обычными переменными оболочки и переменными окружения.

В этом разделе мы рассмотрим другие стороны работы с переменными, особенно важные для оболочкового программирования. Точнее, поговорим об их области определения и о некоторых переменных, заданных в bash (например, $* или $?), о механизме подстановки параметров для анализа и обработки последовательностей символов, содержащихся в переменных и, наконец, рассмотрим вопросы ввода переменных в программах оболочки.

Область определения переменных

Чтобы понять тонкости применения переменных при выполнении программ оболочки, необходимы базовые знания о механизмах запуска команд и программ оболочки.

Для выполнения команды или программы bash создает новый процесс с собственным PID-номером (такой номер используется внутри системы Linux для идентификации процесса и управления им). Новому процессу сообщаются только те оболочковые переменные, которые были объявлены как переменные окружения (`export` или `declare -x`). Если команда запускается в приоритетном режиме, bash продолжает работать на фоне этой команды, ожидая, пока она будет выполнена. В противном случае обе программы (то есть bash и запущенная в фоновом режиме команда) выполняются параллельно.

Особый случай — запуск оболочковой программы. Программы оболочки выполняются отнюдь не в работающей оболочке, а в специально запускаемом для этого командном подпроцессоре. В таком случае одновременно работают два образца bash — один для интерпретации команд, а другой для выполнения программ оболочки. Отдельные командные подпроцессоры требуются оболочке для того, чтобы можно было параллельно выполнять несколько программ оболочки, чтобы они при этом не влияли друг на друга (даже если работают в фоновом режиме).

Использование командных подпроцессоров в особенности отражается на управлении переменными в тех случаях, когда каждый подпроцессор (или оболочка) обладает собственным набором переменных. При запуске любой другой программы ей сообщаются лишь те переменные интерактивной оболочки, которые были объявлены как переменные окружения. Переменные оболочки и командного подпроцессора работают полностью независимо друг от друга, то есть при внесении изменений в один набор переменных это никак не отражается на другом наборе.

Иногда может потребоваться объявить в работающей оболочковой программе новые переменные либо изменить имеющиеся переменные. Чтобы это получилось, можно выполнять оболочковые программы и в рамках уже запущенной оболочки bash, не используя командный подпроцессор. Для этого нужно поставить перед именем файла оболочковой программы точку или пробел. Именно так кратко записывается оболочковая команда `source`.

Вот пример: вы хотите написать оболочковую программу, которая бы дополняла переменную `PATH` путем к текущему каталогу. Нужная вам программа `addpwd` очень проста:

```
#! /bin/sh
# оболочковая программа addpwd добавляет путь к текущему каталогу
#
PATH=$PATH":"$(pwd)
```

Итак, теперь в переменной `PATH` сохраняется ее прежнее содержимое, двоеточие и результат выполнения команды `pwd` — последняя часть сохраняется благодаря подстановке команд. Следующий контрольный прогон показывает, что содержимое переменной `PATH` текущей оболочки изменится лишь тогда, когда вы поставите перед запускаемой командой `addpwd` точку (в командном подпроцессоре, запуска-

емом при первом обращении к addpwd, PATH также изменяется, но эти изменения действуют, пока выполняется addpwd).

```
user$ echo $PATH
/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin
user$ addpwd
user$ echo $PATH
/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin
user$ . addpwd
user$ echo $PATH
/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin:/home/user
```

Переменные, задаваемые оболочкой

Программы оболочки могут обращаться и к некоторым переменным, заранее заданным bash. Эти переменные нельзя изменять в ходе присваивания, их можно только считывать. Имя переменной состоит из различных специальных символов. В табл. 6.5 переменные даются уже с предшествующим символом $.

Таблица 6.5. $-переменные

Переменная	Значение
$?	Возвращаемое значение последней команды
$!	Номер PID последнего запущенного фонового процесса
$$	PID текущей оболочки
$0	Имя файла, содержащего только что выполненный файл оболочки (или имя символьной ссылки, указывающей на файл)
$#	Количество параметров, переданных программе оболочки
От $1 до $9	Параметр от 1 до 9
$* или $@	Совокупность всех переданных параметров

Еще несколько замечаний относительно работы с переменными: $0-$9, $# и $* служат для интерпретации параметров, передаваемых программе пакетной обработке. Эти переменные могут использоваться практически с любым из сценариев, показанных в этой главе.

В связи с интерпретацией параметров интересна команда bash под названием shift. Она как будто перемещает переданные параметры от переменной $0 до переменной $9. Если вы выполните shift 9, то первые девять параметров, переданных программе, будут потеряны, но вы легко сможете запросить следующие девять. Если указать shift без дополнительных параметров, то список сдвинется на один параметр.

Переменную $? можно использовать для определения условий, чтобы поставить дальнейшее выполнение программы в зависимость от результата последней команды. В принципе команду можно прямо задать как условие, указав ее таким образом в if. Достоинство переменной $? заключается в том, что она избавляет от работы с длинными и непонятными командами.

Переменная $$ содержит PID (идентификационный номер процесса). Это числовое значение применяется внутри системы Linux для управления процессами.

PID является уникальным, то есть во всей системе гарантированно отсутствует второй процесс с таким же номером, поэтому такое значение отлично подходит для создания временного файла. Например, с помощью `ls > tmp.$$` можно сохранить список всех файлов в файле `tmp.`*nnn*. Даже если такой же пакетный файл одновременно работает на другом терминале, в двух оболочках процессы будут иметь разные идентификационные номера, что исключает возникновение конфликта на уровне имен.

Массивы

Кроме обычных переменных, bash также различает массивы. Вплоть до версии 3 индекс должен был быть числом. Обратите внимание, что синтаксис для доступа к *n*-ному элементу отличается от синтаксиса, принятого в C.

```
x=()              # Определение пустого массива
x[0]='a'          # Присваивание элементов массива
x[1]='b'
x[2]='c'
x=('a' 'b' 'c')   # Краткий вариант записи четырех предыдущих строк
echo ${x[1]}      # Считывание элемента массива
echo ${x[@]}      # Считывание всех элементов массива
```

Для программистов исключительно важны массивы ассоциативных элементов, которые стали поддерживаться в bash, начиная с версии 4.0. Не забудьте, что массив сначала требуется специально объявить как ассоциативный с помощью `declare -A`. В противном случае система сочтет, что это обычный массив. Тогда последовательности символов, содержащиеся в индексе, будут интерпретированы как `0`, и у вас получится обычный массив, состоящий из одного-единственного элемента (`Index 0`).

```
declare -A y       # Определение пустого массива ассоциативных элементов
y[abc]=123         # Присваивание элемента ассоциативного массива
y[efg]=xxx
y=( [abc]=123 [efg]=xxx )# Краткий вариант записи двух предыдущих строк
echo ${y[abc]}           # Считывание одного элемента массива
```

Еще одно важное нововведение версии 4 заключается в том, что с помощью команды `mapfile` можно построчно преобразовать текстовый файл в элементы обычного массива:

```
mapfile z < текстовый_файл
```

Подстановка параметров

В bash под подстановкой параметров понимается использование ряда команд, с помощью которых можно обрабатывать последовательности символов, сохраненные в переменных. Обратите внимание, что перед именем переменной необходимо ставить символ `$`. Кроме того, если из переменной должен считываться эталонный образец, также необходимо использовать символ `$`.

- `${var:-default}` — если переменная пуста, такое выражение возвращает в качестве результата установку, заданную по умолчанию. В противном случае возвращается содержимое переменной. Сама переменная не изменяется.
- `${var:=default}` — аналогично предыдущему, однако если до операции переменная была пуста, ее содержимое изменяется.
- `${var:+new}` — если переменная была пуста, то она остается пустой. Если же переменная уже определена, то ее предыдущее содержимое заменяется новым. Тогда выражение возвращает новое содержимое переменной.
- `${var:? Error_message}` — если переменная пуста, то выводится имя переменной и сообщение об ошибке. Кроме того, программа оболочки завершает работу. В противном случае выражение возвращает содержимое переменной.
- `${#var}` — в качестве результата возвращает количество символов, сохраненных в переменной (0, если переменная пуста). Переменная не изменяется.
- `${var#pattern}` — сравнивает начало переменной с заданным шаблоном. Если удается опознать шаблон, то выражение возвращает содержимое переменной за исключением самого краткого фрагмента текста, хранящего поисковый шаблон. Если же соответствий шаблону найти не удается, то возвращается содержимое переменной целиком. В поисковом шаблоне можно использовать подстановочные символы, с которыми мы познакомились, обсуждая образование имен файлов (`* ? [abc]`). Переменная в любом случае остается неизменной:

```
user$ dat=/home/mk/buch/buch.tar.gz
user$ echo ${dat#*/}
home/mk/buch/buch.tar.gz
user$ echo ${dat#*.}
tar.gz
```

- `${var##pattern}` — аналогично предыдущему, но теперь удаляется самый крупный фрагмент текста, содержащий поисковый шаблон:

```
user$ dat=/home/mk/buch/buch.tar.gz
user$ echo ${dat##*/}
buch.tar.gz
user$ echo ${dat##*.}
gz
```

- `${var%pattern}` — аналогично `${var#pattern}`, но в данном случае сопоставление образцов производится в конце содержимого переменной. В конце переменной удаляется кратчайший фрагмент, содержащий поисковый шаблон. Сама переменная не изменяется:

```
user$ dat=/home/mk/buch/buch.tar.gz
user$ echo ${dat%/*}
/home/mk/buch
user$ echo ${dat%.*}
/home/mk/buch/buch.tar
```

- `${var%% pattern}` — аналогично предыдущему, но теперь удаляется самый крупный фрагмент текста, содержащий поисковый шаблон и находящийся в конце переменной:

```
user$ dat=/home/mk/buch/buch.tar.gz
user$ echo ${dat%%/*}
-- вывод отсутствует --
user$ echo ${dat%%.*}
/home/mk/buch/buch
```

- `${var/find/replace}` — замена первого совпадения с шаблоном `find` посредством `replace`:

```
user$ x='abcdeab12ab'
user$ echo echo ${x/ab/xy}
xycdeab12ab
```

- `${var//find/replace}` — замена всех совпадений с шаблоном `find` посредством `replace`:

```
user$ x='abcdeab12ab'
user$ echo echo ${x//ab/xy}
xycdexy12xy
```

- `${!var}` — возврат содержимого переменной, имя которой в виде последовательности символов содержится в `var`:

```
user$ abc="123"
user$ efg=abc
user$ echo ${!efg}
123
```

Считывание переменных с помощью read

С помощью bash-команды `read` можно обрабатывать пользовательский ввод. Как правило, для этого сначала с помощью `echo` задается краткий текст, сообщающий пользователю, какой ввод ожидается (например, `y/n`, числовое значение и т. д.). При этом целесообразно использовать параметр `-n`, позволяющий вводить данные сразу же после текста `echo`, а не со следующей строки. При выполнении последующей команды `read` bash ожидает от пользователя ввода строки и нажатия клавиши Enter.

Возьмем программу, в которой цикл `while` выполняется до тех пор, пока в переменной `a` не окажется последовательность символов, удовлетворяющая определенным условиям. В следующем примере мы видим, как работает эта программа:

```
user$ readvar
Введите число: a
Неверный ввод, повторите ввод, пожалуйста
Введите число: 12
12
```

После ввода информации с помощью `read` все содержимое переменной удаляется посредством подстановки параметров, если последовательность включает любые символы, кроме цифр, знаков «минус» и пробелов. Правда, эта система контроля не идеальна (в соответствии с ней допускается как последовательность символов `12-34-5`, так и `1234`), но все же весьма эффективна. О циклах `while` будет рассказано в подразделе «Циклы while» раздела 6.11.

```
#! /bin/sh
# Пример readvar: считывание числового значения
a=                              # Удаление содержимого переменной a
while [ -z "$a" ]; do
 echo -n "Введите число: "
 read a  a=${a##*[^0-9,' ',-]*} # Удаление последовательностей символов,
                                # содержащих любые символы, кроме 0-9,
                                # знака "минус" и пробела
 if [ -z "$a" ]; then
  echo "Неверный ввод, повторите ввод, пожалуйста"
 fi
done
echo $a
```

6.11. Условные переходы и циклы в сценариях bash

Условные переходы в программах оболочки создаются с помощью команд `if` и `case`. В то время как `if` лучше подходит для обычных операций выбора, `case` предназначен для анализа последовательностей символов (сопоставления образцов).

If-условные переходы

В файле `iftst` с помощью `if`-запроса мы проверяем, были ли переданы два параметра. Если этого не произошло, выводится сообщение об ошибке. Программа завершается командой `exit` с возвращаемым значением, неравным нулю (индикатор ошибок). В противном случае содержимое обоих параметров выводится на экран.

```
#! /bin/sh
# Пример iftst
if test $# -ne 2; then
 echo "Команде должно быть передано ровно два параметра!"
 exit 1
else
 echo "Параметр 1: $1, Параметр 2: $2"
fi
```

Короткий тестовый запуск показывает, как работает программа:

```
user$ iftst a
```

Команде должно быть передано ровно два параметра!

```
user$ iftst a b
Параметр 1: a, Параметр 2: b
```

Критерием для условного перехода является возвращаемое значение последней команды перед `then`. Если эта команда возвращает значение `0`, то условие выполняется. Если `then` указывается в той же строке, что и предыдущая команда (без перехода на следующую), то команда должна завершаться точкой с запятой.

ВНИМАНИЕ

Обратите внимание, что значение истинности (true) в bash равно 0, а ложности (false) — не равно 0. В большинстве других языков программирования значения прямо противоположны! Команды, завершаемые без ошибок, возвращают значение 0. Любое значение, не равное 0, указывает на ошибку. Некоторые команды возвращают различные значения в зависимости от типа ошибки.

В предыдущем примере условие было создано с помощью команды `test` оболочки bash. При этом оператор `-ne` означает «не равно» (not equal). Команда `test` применяется всякий раз, когда необходимо сравнить друг с другом две последовательности символов или два числа, когда необходимо проверить, существует ли файл, и т. д. Эта команда будет описана в следующем подразделе.

Предыдущую программу можно было сформулировать иначе: вместо `test` может использоваться краткий вариант в квадратных скобках. При этом перед названием команды [и после него] нужно ставить пробел.

Кроме того, из структуры `if` можно вычленить вторую команду `echo`, так как из-за наличия команд `exit` все строки кода, следующие за `fi`, будут выполняться только после того, как будет реализовано условие.

```
#! /bin/sh
# Пример iftst, 2. вариант
if [ $# -ne 2 ]; then
 echo " Команде должно быть передано ровно два параметра!"
 exit 1
fi
echo "Параметр 1: $1, Параметр 2: $2"
```

Формулирование условий с помощью test

В bash невозможно прямо задавать условия, например, для сравнения переменной и значения. Во-первых, вся архитектура bash основана на том, что все действия осуществляются с помощью команд, построенных по одинаковому образцу, во-вторых, специальные символы < и > уже закреплены для выполнения других функций. Поэтому для формулирования условий в циклах и точках перехода в bash необходимо использовать команду `test`. (Между прочим, `test` является самостоятельной командой и существует не только в bash. Эта команда была интегрирована в bash, чтобы увеличить скорость обработки данных.)

Команда `test` возвращает значение `0` (истинно), если условие выполняется, или `1` (ложно), если не выполняется. Для сокращения печатной работы предусмотрен краткий вариант записи в квадратных скобках.

Команда `test` используется для решения задач двух классов: для сравнения двух чисел, для сравнения последовательностей символов и для того, чтобы узнать, существует ли файл и проявляет ли он определенные свойства. В следующих примерах показаны некоторые возможные варианты применения этой команды.

- `test "$x"` — проверяет, занят ли `x` (то есть, если в последовательности символов содержится 0 символов, имеем «ложно»; в противном случае «истинно»).
- `test $x -gt 5` — проверяет, имеет ли переменная `x` числовое значение больше 5. Если `x` не содержит числа, выводится сообщение об ошибке. Вместо `-gt` (больше)

могут также использоваться следующие операторы сравнения: `-eq` (равно), `-ne` (не равно), `-lt` (меньше), `-le` (меньше или равно) и `-ge` (больше или равно).

- `test -f $x` — проверяет, существует ли файл с именем, указанным в `x`.

Если необходимо интерактивно выполнять команду `test` в оболочке, после нее нужно считать значение переменной `$?` (возвращаемое значение последней команды) с помощью `echo`:

```
user$ a=20
user$ test $a -eq 20; echo $?
0
user$ test $a -gt 20; echo $?
1
```

Case-условные переходы

Конструкции `case` вводятся ключевым словом `case`, за которым следует параметр, предназначенный для анализа (обычно это переменная). После ключевого слова `in` можно указать несколько возможных шаблонов строки, с которыми будет сравниваться параметр. При этом могут использоваться те же джокерные символы, что и при работе с именами файлов. Шаблон завершается круглой скобкой, то есть, например, `--*)`. Это делается для распознавания таких последовательностей, которые начинаются с двух знаков «минус». Несколько шаблонов можно отделять друг от друга символом `|`. В таком случае проверяются оба шаблона. Например, `*.c|*.h)` служит для распознавания файлов `*.c` и `*.h` в одном и том же ветвлении программы.

Команды, идущие вслед за скобками, должны завершаться двумя точками с запятой. Если потребуется переход `else`, то в качестве последнего шаблона необходимо указать * — такому шаблону будут соответствовать все последовательности символов. При обработке конструкции с `case` учитывается только первый переход, в котором параметр соответствует указанному шаблону.

В следующем примере `casetst` показано применение `case` для классификации переданных параметров на имена файлов. Цикл для переменной `i` выполняется со всеми параметрами, переданными файлу оболочки. В этом цикле каждый отдельный параметр анализируется с помощью `case`. Если параметр начинается с дефиса (`-`), то он находится в конце переменной `opt`, в противном случае — в конце `dat`.

```
#! /bin/sh
# Пример casetst
opt=        # Удаление opt и dat
dat=

for i do    # Цикл для всех переданных параметров
 case "$i" in
  -* ) opt="$opt $i";;
  * ) dat="$dat $i";;
 esac
done        # Конец цикла
echo "Options: $opt"
echo "Files: $dat"
```

Тестовый запуск файла оболочки на практике показывает, как работает этот простой оператор выбора. Параметры, переданные по порядку без сортировки, подразделяются на имена файлов и параметры:

```
user$ casetst -x -y dat1 dat2 -z dat3
Options: -x -y -z
Files: dat1 dat2 dat3
```

По тому же принципу условные переходы case могут использоваться для классификации определенных расширений файлов (путем указания поискового шаблона *.abc). Если вы хотите плотнее заняться case-анализом, посмотрите файл оболочки /usr/bin/gnroff. В нем дается синтаксис параметров, передаваемых nroff, в виде, понятном родственной команде groff.

For-циклы

Циклы в bash создаются с помощью трех команд. Команда for осуществляет цикл со всеми элементами указанного списка. Команда while выполняет цикл до тех пор, пока указанное условие не перестанет выполняться. Команда until, наоборот, осуществляет цикл до тех пор, пока указанное условие не будет выполнено. Все три цикла можно досрочно завершить командой break. Команда continue пропускает оставшуюся часть тела цикла и запускает цикл заново.

В первом примере переменной i по очереди присваиваются последовательности символов a, b и c. В теле цикла между do и done выводится содержимое переменной. Обратите внимание, что в конце списка, а также в конце команды echo необходимо поставить точку с запятой. Отказаться от этих точек с запятой можно лишь тогда, когда ввод разделен на несколько строк (часто такое случается в сценарных файлах).

```
user$ for i in a b c; do echo $i; done
a
b
c
```

Эквивалентная многострочная формулировка вышеуказанной команды в сценарном файле выглядела бы так:

```
#! /bin/sh
for i in a b c; do
 echo $i
done
```

Список для for может быть построен с использованием джокерных символов для имен файлов или с использованием конструкций вида {..}, с помощью которых создаются последовательности символов. В следующем примере все файлы *.tex копируются в *.tex~. (В Linux/UNIX тильда (~) в конце имени файла обычно означает резервную копию. При работе с командой cp выражение $file всякий раз ставится в кавычках, чтобы имена файлов, содержащие пробелы, обрабатывались правильно.)

```
user$ for file in *.tex; do cp "$file" "$file~"; done
```

Если циклы for создаются без in ..., то переменные циклов получают по порядку все параметры, переданные при вызове (то есть это соответствует in $*). Пример такого цикла приводится при описании case.

Но когда в примере с case передаются имена файлов, содержащие пробелы, без проблем не обходится. Bash интерпретирует пробелы как разделительные знаки и обрабатывает части имени файла отдельно. Чтобы справиться с такой проблемой, применяется следующая конструкция:

```
#!/bin/bash
# Цикл проходит через все параметры,
# правильно справляется с пробелами в именах файлов
for i in "$@"; do
 ls -l "$i"
done
```

Циклы while

В следующем примере переменной i присваивается значение 1. Потом значение переменной, находящейся в теле цикла между do и done, при каждом выполнении цикла увеличивается на 1, пока не будет превышено значение 5. Обратите внимание, что условия должны указываться в квадратных скобках, как это делалось с условными переходами if, с командой test или с ее сокращенным вариантом.

```
user$ i=1; while [ $i -le 5 ]; do echo $i; i=$[$i+1]; done
1
2
3
4
5
```

Следующий цикл обрабатывает все имена файлов, получаемые после выполнения команды ls *.jpg:

```
ls *.jpg | while read file
do
 echo "$file"
done
```

Циклы until

Единственное отличие между циклами while и until заключается в том, что условие формулируется с противоположной логикой. Следующая команда эквивалентна вышеуказанному циклу while. При этом для формулировки условия i>5 применяется оператор -gt (больше).

```
user$ i=1; until [ $i -gt 5 ]; do echo $i; i=$[$i+1]; done
1
2
3
4
5
```

6.12. Справка по важнейшим специальным символам bash

И при вводе команд, и при программировании оболочки для выполнения различных действий применяется множество специальных символов. В табл. 6.6 приведены все специальные символы, рассмотренные в этой главе.

Таблица 6.6. Специальные символы, используемые в bash

Символ	Значение
;	Отделение команд друг от друга
:	Команда оболочки, ничего не делает
.	Запуск оболочки без собственного командного подпроцессора (.file соответствует исходному файлу)
#	Ввод комментария
#!/bin/sh	Идентификация оболочки, в которой будет выполняться программа
&	Выполнение команды в фоновом режиме (com &)
&&	Выполнение одной команды в зависимости от результата другой (com1 && com2)
&>	Переадресация стандартного вывода и ошибок (соответствует >&)
|	Создание программных каналов (com1 | com2)
||	Выполнение одной команды в зависимости от результата другой (com1 || com2)
*	Джокерный символ для имен файлов (любое количество символов)
?	Джокерный символ для имен файлов (любой символ)
[abc]	Джокерный символ для имен файлов (любой символ из abc)
[expression]	Сокращенный вариант записи test expression
(...)	Выполнение команд в той же оболочке ((com1; com2))
{...}	Группирование команд
{ , , }	Объединение нескольких последовательностей символов (a{1,2,3} → a1 a2 a3)
{a..b}	Объединение нескольких последовательностей символов (b{4..6} → b4 b5 b6)
~	Сокращенное обозначение домашнего каталога
>	Переадресация вывода в файл (com > file)
>>	Переадресация вывода и добавление его в существующий файл
>&	Переадресация стандартного вывода и ошибок (соответствует &>)
2>	Переадресация стандартного вывода ошибок
<	Переадресация ввода из файла (com < file)
<< end	Переадресация ввода из активного файла до завершения
$	Обозначение переменных (echo $var)
$!	Номер PID последнего процесса, запущенного в фоновом режиме
$$	PID актуальной оболочки
$0	Имя выполняемого в данный момент сценарного файла оболочки
$1–$9	Первые девять параметров, переданных команде
$#	Количество параметров, переданных программе оболочки

Символ	Значение
$* или $@	Совокупность всех переданных параметров
$?	Возвращаемое значение последней команды (0=OK или номер ошибки)
$(...)	Подстановка команд (echo $(ls))
${...}	Различные специальные функции для обработки последовательностей символов
$[...]	Арифметические вычисления (echo $[2+3])
"..."	Предотвращение интерпретации большинства специальных символов
'...'	Предотвращение интерпретации всех специальных символов
`...`	Подстановка команд (echo `ls`)

7 Управление файлами

В данной главе описаны способы управления файлами. Будут подробно рассмотрены следующие темы:

- файлы, каталоги и ссылки;
- копирование, перемещение, удаление и поиск файлов и каталогов;
- запись CD и DVD;
- права доступа к файлам (включая список контроля доступа);
- структура каталогов в Linux;
- файлы-устройства.

Администрирование файловой системы. В главе 14 обсуждается администрирование файловых систем. Данная глава в определенной степени представляет собой продолжение этой главы, но там будут рассматриваться вопросы, более важные не для пользователей, а для системных администраторов, а именно:

- какие файловые системы существуют;
- как файлы интегрируются в систему [`/etc/fstab`, `mount`-параметры];
- как можно применять систему программ массива независимых жестких дисков;
- что такое LVM;
- как можно зашифровать целую файловую систему.

Резервное копирование. Вопросы резервного копирования будут рассмотрены в главе 25. Там я расскажу не о пользовательских интерфейсах, обеспечивающих резервное копирование, а о богатом наборе команд, позволяющих сжимать файлы, объединять их в архивы, синхронизировать, шифровать и т. д.

7.1. Работа с файлами и каталогами

Коротко перечислю важнейшие фактические сведения об именах файлов.

- В ОС Linux имя файла должно быть не длиннее 255 символов.
- Имена файлов чувствительны к регистру!
- В именах файлов допускается указание международных символов, однако при использовании различных кодировок могут возникать проблемы (например,

если разные кодировки одновременно применяются в одной и той же сети). На протяжении уже более пяти лет почти во всех дистрибутивах Linux стандартной кодировкой считается UTF-8.

Для ядра Linux имя файла — это просто байтовая последовательность, в которой не может присутствовать символ или код 0. Интерпретация этой байтовой последовательности зависит от применяемой кодировки.

- В имени файла может содержаться сколько угодно точек. Имя файла `README.bootutils.gz` является совершенно тривиальным и означает, что мы имеем дело с файлом `README` по теме «Загрузочные утилиты».
- Файлы, имена которых начинаются с точки, считаются скрытыми (см. подраздел «Скрытые файлы» этого раздела). Как правило, скрытые файлы не отображаются в программе `ls` и в различных файловых менеджерах.
- Имена файлов, которые невозможно однозначно опознать как таковые после ввода команды (в частности, имена файлов, содержащие пробелы) должны даваться в кавычках (например: `"a b"`).

Размер файлов в современных версиях Linux практически ничем не ограничен и в зависимости от файловой системы может исчисляться терабайтами.

Каталоги

Дерево каталогов. Дерево каталогов в Linux начинается с корневого каталога `/`. Указывать диск, например, `C:` в Linux невозможно. В этой книге все остальные каталоги считаются *подчиненными*. Таким образом, если представить дерево каталогов наглядно, то корневой каталог расположен на самой его верхушке. В некоторых книгах используется прямо противоположная номенклатура, больше напоминающая дерево (корни внизу, ветви вверху), но это не соответствует обычному значению терминов.

Одна из наиболее серьезных проблем в начале работы с Linux/UNIX заключается в том, чтобы найти определенный файл в широко разветвленной системе каталогов. Эта проблема рассматривается в разделе 7.8.

Личный каталог. При работе с текстовыми консолями (терминалами) после входа в систему вы сразу же автоматически оказываетесь в так называемом личном, или домашнем, каталоге. В вашем распоряжении все находящиеся в нем файлы и подкаталоги. Другие пользователи (кроме того, который обладает привилегиями администратора) не могут ни изменять, ни удалять файлы вашего личного каталога (а при определенных настройках даже не могут их читать).

Личный каталог обычно находится в дереве каталогов Linux по адресу `/home/loginname/` (только личный каталог администратора называется `/root`). Поскольку было бы неудобно каждый раз писать `/home/loginname/`, название собственного домашнего каталога сокращается до символа тильды (~). Кроме того, доступ к личным каталогам других пользователей возможен через запись `~loginname`.

Каталоги . и .. . В каждом каталоге имеется два особых подкаталога, необходимых для формального управления иерархией каталогов. Каталог с именем, начинающимся с `.`, представляет собой ссылку на актуальный каталог, а каталог

с именем, которое начинается с .., — на каталог, расположенный уровнем выше (табл. 7.1).

Таблица 7.1. Специальные символы, используемые при работе с каталогами

Символ	Значение
~	Домашний каталог
.	Текущий каталог
..	Каталог, расположенный на один уровень выше, чем текущий

Обе следующие команды копирования показывают, как можно использовать эти каталоги (другие cp-примеры приводятся далее). Первая команда копирует файл /etc/fstab в актуальный каталог. Если каталог называется /home/name, то новый файл будет иметь имя /home/name/fstab.

```
user$ cp /etc/fstab .
```

Во втором примере мы сначала активизируем с помощью команды cd каталог ~/linux8. Затем команда копирования cp создает резервную копию файла fileuse.tex (в котором содержится текст этой главы). Резервная копия имеет имя ~/fileuse.tex.bak.

```
user$ cd ~/linuxbuch
user$ cp fileuse.tex ../fileuse.tex.bak
```

Если домашний каталог называется /home/name, то вы сейчас создали резервную копию /home/name/linuxbuch/fileuse.tex. Следовательно, полное название резервной копии: /home/name/fileuse.tex.bak.

Простейшие команды для работы с каталогами. Хотя при работе с KDE и Gnome в вашем распоряжении есть современные файловые менеджеры, опытные пользователи Linux охотнее работают с текстовыми командами. В табл. 7.2 приведены самые важные из таких команд.

Таблица 7.2. Команды для работы с каталогами

Команда	Функция
cd	Смена актуального каталога
cp	Копирование файлов
less	Постраничный показ текстовых файлов
ls	Показ всех файлов каталога
mkdir	Создание нового каталога
mv	Перемещение файлов или изменение их имен
rm	Удаление файлов
rmdir	Удаление каталогов

Переход между каталогами с помощью команды cd. Команда cd позволяет перейти в другой каталог. Команда cd – возвращает вас в каталог, который был активен непосредственно перед этим, cd .. переводит в подкаталог, а команда cd без дополнительных параметров позволяет перейти в личный каталог.

```
user$ cd /etc/samba
```

Переход в другой каталог с помощью j. В некоторых дистрибутивах для перехода в другой каталог используется команда `j` из пакета `autojump`. Как правило, сначала требуется установить этот пакет. Кроме того, `autojump` запоминает, в каких каталогах вы работаете чаще всего. При необходимости можно отобразить соответствующие статистические данные с помощью команды `jumpstats`.

Если вы хотите вернуться в каталог, в котором работали ранее, выполните команду `j abc`, где `abc` — первые три буквы имени каталога. (Можно обойтись без ввода пути к каталогу, который бывает достаточно длинным.) Если в системе находится несколько подходящих под такой запрос каталогов, `j` переводит вас в тот из них, с которым вы в последнее время работали чаще всего. Кроме того, вы можете выбрать один из предложенных на выбор каталогов с помощью клавиши Tab.

На сайте Autojump (https://github.com/joelthelion/autojump/wiki) `j` охарактеризована как «самообучаемый» вариант команды `cd`. Точнее не скажешь. Чтобы привыкнуть к `j`, требуется совсем немного времени — а после этого все получается само собой.

Но команда `j` в любом случае не заменяет, а лишь дополняет `cd`. Автозавершение пути с помощью клавиши Tab функционирует только с теми каталогами, которые уже занесены в базу данных Autojump. То есть при вводе `cd /e` и нажатии Tab значение `/e`, как правило, дополняется до `/etc/`. А в случае с `j` аналогичное автозавершение начнет происходить лишь после того, как `/etc` окажется в базе данных Autojump. Иными словами: чтобы перейти в каталог, еще не известный Autojump, лучше пользоваться командой `cd`. (Попасть в желаемый каталог можно и с помощью `j`, но в описанном случае имя такого нового каталога приходится полностью вводить самостоятельно.) Жаль, что `j` не может полностью заменить `cd`!

Перечисление файлов. Команда `ls` возвращает список всех файлов каталога. Если вы также хотите увидеть скрытые файлы, то задайте дополнительный параметр `-a`. Если вас интересует не только имя файла, но и другая информация о нем, например размер, владелец и т. д., то вам поможет параметр `-l`. По умолчанию вывод информации по `ls` производится в алфавитном порядке. Чтобы сортировать файлы по времени внесения изменений, размеру или расширению, используйте параметры `-t`, `-S` или `-X`. Параметр `-r` позволяет отсортировать информацию в обратном порядке. Следующая команда показывает все файлы с расширением TEX в каталоге linuxbuch, расположенные в порядке уменьшения размера:

```
user$ ls -l -S linuxbuch/*.tex
...
-rw-r--r-- 1 kofler kofler 301132012-05-1109:09 linuxbuch/intro.tex
-rw-r--r-- 1 kofler kofler 631732012-01-2908:05 linuxbuch/kde.tex
-rw-r--r-- 1 kofler kofler 764982012-06-0815:43 linuxbuch/kernel.tex
...
```

Приведу несколько замечаний по интерпретации вывода команды `ls`: десять первых символов в начале строки указывают тип файла и биты доступа. В описание типа файла входят дефис (`-`) для обычного файла, `d` для каталога (directory), `b` или `c` для файла устройства (block или char) или `l` для символьной ссылки. Три следующих символа (`rwx`) указывают, что пользователи могут делать с файлом: читать, записывать в него информацию или выполнять его. Кроме того, сообщается

аналогичная информация для членов группы, а также для других пользователей системы. Число, следующее за десятью символами, означающими тип и уровень доступа, говорит о том, сколько жестких ссылок указывает на файл. В следующих столбцах указано, к какой группе относится файл и кто его владелец (здесь в обоих случаях kofler), размер файла, дата его последнего изменения и, наконец, имя файла.

В большинстве дистрибутивов команда ls сконфигурирована так, что файлы и каталоги распределяются по типу и окрашиваются разными цветами в зависимости от типа. Если в вашем дистрибутиве это не так, подобного эффекта можно достичь добавлением дополнительного параметра --color.

Обычно ls учитывает только файлы, находящиеся в каталоге, открытом в настоящий момент. Если вы хотите учесть также файлы, которые располагаются в подкаталогах, используйте параметр -R. Он, кстати, используется и со многими другими командами.

Следующая команда позволяет перечислить все без исключения файлы, находящиеся во всех подкаталогах (в том числе скрытые каталоги и файлы). Как правило, этот список достаточно длинный. Из него команда | less передает результат от ls к less, так что теперь вы можете пролистывать полученный программный вывод.

```
user$ ls -lR | less
```

Копирование файлов. Команда cp *имя1 имя2* копирует файл *имя1*. Копия называется *имя2*. Чтобы скопировать несколько файлов, вызовите команду вида cp *имя1 имя2 ... целевой каталог*. Следующие команды сначала переводят в активное состояние каталог linuxbuch, затем создают подкаталог bak, а потом копируют туда все текстовые TEX-файлы.

```
user$ cd linuxbuch
user$ mkdir bak
user$ cp *.tex bak/
```

Копирование каталогов. Чтобы копировать целые каталоги вместе со всем их содержимым, используйте команду cp -r. Параметр -r обеспечивает рекурсивную обработку всего содержимого исходного каталога (в том числе, его скрытых файлов). Если вы хотите сохранить при копировании информацию о правах и времени доступа, используйте вместо -r параметр -a.

Вопрос о том, копируется ли сам исходный каталог либо только его содержимое, является довольно тонким. Если целевой каталог уже известен, то в нем создается подкаталог sourcefolder (то есть исходный каталог), и туда копируется все содержимое исходного каталога. Если же целевого каталога еще нет, то он создается. В этом случае в новоиспеченный целевой каталог копируется только содержимое исходного каталога, но не сам этот каталог. Разница понятна из следующих примеров:

```
user$ mkdir test
user$ touch test/a user$ mkdir test/b user$ touch test/b/c user$ mkdir ziel1
user$ cp -r test ziel1      (Каталог ziel1/ уже существует)
user$ ls ziel1
```

```
test
user$ cp -r test ziel2      (Каталог ziel2/ еще не существует)
user$ ls ziel2
a b
```

Удаление файлов и каталогов. Команда `rm` позволяет безвозвратно удалить указанный файл. Как правило, `rm` применяется для удаления файлов, а не каталогов. Для удаления каталогов предусмотрена команда `rmdir` *`каталог`*, которая, правда, работает лишь в том случае, если указанный каталог пуст. На практике для удаления каталогов обычно применяется команда `rm` с параметром `-rf`. Это означает, что все каталоги и находящиеся в них подкаталоги и файлы рекурсивно удаляются без запроса о подтверждении удаления. Понятно, что команда `rm -rf` очень опасна!

```
userrm -rf linuxbuch-bak/ (Удаление папки Backup-Verzeichnis)
```

Как узнать, сколько памяти нужно для размещения всех файлов и каталогов

С помощью команды `ls -l` можно выяснить, насколько велик файл. Однако зачастую требуется знать, сколько места занимают все файлы, находящиеся в каталоге, сколько свободного места еще есть на диске и т. д. Для этого вам пригодятся две команды: `df` и `du`.

Узнаем, сколько свободного места на диске. Команда `df` показывает для всех сегментов файловой системы, или носителей данных, сколько всего места есть на этих носителях и сколько еще свободно. Параметр `-h` позволяет отобразить все данные о доступном дисковом пространстве в удобочитаемых числах, в килобайтах, мегабайтах или гигабайтах (а не блоками по 1 Кбайт, как это задано по умолчанию). Команда `df` также показывает различные файловые системы, необходимые только для внутреннего управления ресурсами, а не для сохранения обычных файлов.

```
user$ df -h
Файл. система  Размер  Использ.  Дост.  Использ.%  Прикреплено к
/dev/sda3         14G     4,7G   8,5G         36%  /
/dev/sda2        942M      47M   849M          6%  /boot
/dev/sda6         28G     7,7G    19G         30%  /home
...
```

Команда `df` также позволяет установить, в каком сегменте диска физически находится файл. В следующем примере каталог `/home/kofler` находится в сегменте `/dev/sda6`, который, в свою очередь, расположен в дереве каталогов на месте `/home`.

```
user$ df -h /home/kofler/
/dev/sda6         28G     7,7G    19G         30%  /home
```

Установление размера каталога. Команда `du` дает возможность установить, сколько дискового пространства требует данный каталог, в том числе все содержащиеся в нем подкаталоги. Параметр `-h` позволяет вывести результат в удобочитаемой форме, а не в виде блоков по одному килобайту. Параметров для сортировки

результатов команды du не существует (однако в Gnome имеется программа baobag, позволяющая графически представлять размер каталогов; в KDE подобная наглядная возможность предусмотрена в файловом менеджере Konqueror).

```
user$ du -h fotos/2008
74M    fotos/2008/2008-03-ostern
162M   fotos/2008/2008-08-korsika
66M    fotos/2008/2008-11-diverse
...
2,0G   fotos/2008
```

Джокерные символы

При ежедневном обращении с файлами часто требуется обработать целые группы файлов, например все файлы с расширением TEX. Чтобы такие операции были возможны, при вводе команд Linux предусмотрены так называемые джокерные символы (табл. 7.3).

Таблица 7.3. Джокерные символы для имен файлов

Символ	Значение
?	Любой символ
*	Сколько угодно любых символов (а также ни одного)
[abc]	Один из символов, указанных в скобках
[a–f]	Один из символов, относящийся к указанному диапазону
[!abc] или [^abc]	Все символы, указанные в скобках, должны отсутствовать

Символы * и ? Символ ? служит для указания *любого* символа, а символ * — для указания любого количества символов (в том числе ни одного). Пользователи, которые умеют работать с MS-DOS, на первый взгляд могут решить, что отличий от этой системы нет. Однако это впечатление обманчиво.

- Символ * включает практически любые символы, в том числе точки (кроме точек, с которых начинается имя файла). Если вы хотите обработать все файлы, то в Linux потребуется указать *, а не *.*! (Замечания относительно скрытых файлов приводятся ниже.)
- Если указать много джокерных символов, это никак не повлияет на работу Linux. Например, можно искать с помощью выражения *graf* все файлы, в имени которых содержится graf — в том числе grafik.doc, apfelgraf и README.graf.

Символы [] и [!]. Если указания символов * и ? недостаточно, можно ужесточить ограничение, добавив квадратные скобки. Например, [abc] — это подстановочный символ для одной из трех букв: a, b или c. Если в квадратных скобках между двумя буквами или цифрами стоит дефис, то имеется в виду символ «между». Таким образом, [a-f]* охватывает все файлы, начинающиеся с любой из букв между a и f включительно.

Выражение *[_.-]* означает все файлы, в названии которых содержится минимум одна точка, нижнее подчеркивание или специальный символ, а * .[hc] означает все файлы, которые заканчиваются на .c или .h.

Джокерные символы можно применять и при работе с каталогами. Выражение */*.tex означает все TEX-файлы, которые находятся в подкаталогах текущего каталога (только на один уровень ниже, то есть не включая подкаталоги подкаталогов актуального каталога). Выражение /usr/ * bin/* означает все файлы, находящиеся в каталогах /usr/bin и /usr/sbin.

Интерпретацией джокерных символов занимается не команда, вызываемая для обработки, а оболочка, из которой вызывается команда. Оболочка bash, которая используется в Linux чаще всего, кроме уже указанных джокерных символов распознает и множество других специальных символов, оказывающих специфическое влияние при выполнении команды.

Пример. Следующая команда копирует все C-файлы из каталога project в текущий каталог:

```
user$ cp project/*.c .
```

Сложности при использовании джокерных символов

Работа с джокерными символами на первый взгляд кажется проще, чем она есть на самом деле. Если у вас возникают сложности с джокерными символами, проведите несколько экспериментов с командой echo jokerzeichen. Она показывает все имена файлов, охватываемые комбинацией с джокерным символом, и выводит эти имена на экран, не изменяя при этом имен файлов.

Проблема заключается в том, что символ * относится не только к файлам, но и к каталогам, поэтому команда ls* показывает не только все файлы текущего каталога, но и файлы, находящиеся в подкаталогах, которые также охватываются символом *. При использовании команды ls эту проблему можно устранить с помощью параметра -d, но при работе с другими командами он недоступен.

Обработка каталогов с помощью */. Если вы хотите обработать все каталоги (но не файлы), вам поможет джокерная комбинация */. Она охватывает все «файлы», которые содержат ссылку на самих себя как на подкаталог (а такая ситуация возможна только с каталогами). Внутри системы каталоги считаются особой разновидностью файлов, поэтому я ставлю в данном случае кавычки.

```
user$ echo */.
```

Проблемы с *.окончанием. Тот факт, что обработкой джокерных символов занимается не какая-либо программа, а сама оболочка, имеет не только преимущества, но и недостатки. Например, как оказывается, невозможно произвести поиск TEX-файлов в подкаталогах с помощью команды ls -R *.tex (параметр -R для команды ls вызывает рекурсивный поиск по подкаталогам).

Причина этого проста: оболочка расширяет схему *.tex для текущего каталога и передает список найденных файлов к ls. Эта команда выводит информацию по данным файлам. Если у вас нет каталогов с окончанием *.tex, то команда ls завершает работу. В данном случае не поможет даже параметр -R. Рекурсивный поиск будет применен только к тем каталогам, имена которых были переданы в качестве параметра.

В Linux для поиска по файлам предусмотрена гораздо более гибкая команда find. В следующем примере показан список всех TEX-файлов в текущем каталоге и во всех его подкаталогах. Основы работы и примеры по find приводятся в подразделе «Команды find и grep» раздела 7.4.

```
user$ find . -name '*.tex'
```

Переименование файлов. К сожалению, в Linux нельзя переименовать все файлы *.x в файлы *.y с помощью команды mv *.x *.y. Причина такого ограничения такая же, как и в вышеописанных случаях: оболочка заменяет запись *.x списком всех файлов, соответствующих данной схеме. Для *.y соответствующие имена файлов отсутствуют. При этом команде mv передается список из нескольких файлов и выражение *.y — в этом случае mv не знает, что делать с такими аргументами.

Вот конкретный пример: предположим, что в текущем каталоге находятся только файлы markus.x, peter.x и ulrike.x. Если выполнить в данном случае команду mv *.x *.y, то оболочка заменит схему *.x тремя файлами, указанными выше. Кроме того, оболочка не найдет файлов, подходящих для *.y и передаст схему в неизменном виде. Только сейчас будет запущена команда mv. Она получит следующие параметры, с которыми, как и следовало ожидать, будет невозможна любая работа.

```
user$ mv markus.x peter.x ulrike.x *.y
```

Даже если передать команде в качестве списка параметров запись markus.x peter.x ulrike.x markus.y peter.y ulrike.y, мы получим не тот результат, который нужен. Команда mv в принципе не может переименовать сразу несколько файлов. В таком случае или несколько файлов будут перемещены в другой каталог, или будет переименован только один файл.

Переименование с помощью sed. Разумеется, UNIX-эксперты нашли решение и для этой проблемы: в таких случаях они стали применять потоковый редактор sed. Поскольку обслуживать sed достаточно сложно, примеры, подобные приведенному ниже, подходят только при программировании оболочки.

Кратко опишу принцип работы: команда ls возвращает список файлов, которые необходимо переименовать, и передает их sed. Редактор sed образует вместе с командой s (регулярное нахождение и замена) список cp-команд и в свою очередь передает этот список новой оболочке sh, которая, наконец, выполняет команды. Строка, показанная ниже, позволяет скопировать все файлы *.xxx в *.yyy.

```
user$ ls *.xxx | sed 's/\(.*\)\.xxx$/cp & \1.yyy/' | sh
```

Еще одна альтернатива — написать небольшой цикл. С помощью приведенной ниже команды делаются копии всех файлов с расширением *.tex. Эти копии получают окончание tex ~ (окончание ~ часто применяется для обозначения резервных копий):

```
user$ for i in *.tex; do cp $i $i~; done
```

Скрытые файлы

Файлы, имя которых начинается с точки, в Linux являются скрытыми, поэтому символ * не позволяет учесть все файлы, находящиеся в каталоге: скрытые файлы

(часто ими являются файлы конфигурации, которые должны оставаться невидимыми) игнорируются.

И если вы думаете, что можете включить в выборку все скрытые файлы, поставив подстановочный символ .*, вас ожидает серьезная проблема: таким образом в выборку будут включены не только невидимые файлы, имена которых начинаются с ., но и каталоги, названия которых начинаются с . и .. (то есть текущий каталог и каталог, находящийся в иерархии на одну ступень выше). Если определенная команда сможет изменить целые каталоги, то последствия могут оказаться катастрофическими.

Эту проблему можно обойти, задав схему поиска .[!.]*. Она охватывает все имена файлов, первым символом в которых является точка, далее следует минимум один символ, не являющийся точкой, а затем любое количество символов (в том числе ни одного).

```
user$ echo .[!.]*
```

С командой ls можно использовать параметр -a, который позволяет найти все файлы в данном каталоге (включая скрытые). Такой метод не допускает использования масок (например, *rc*). Параметр -a работает только в том случае, когда ls сама ищет себе файлы, а не передает выполнение этой задачи оболочке.

В данном случае универсальным вариантом является только команда find. Следующая команда находит все скрытые файлы в текущем каталоге:

```
user$ find -maxdepth 1 -type f -name '.*'
```

Особые виды файлов (файлы-ссылки, файлы-устройства)

Кроме обычных, в Linux есть специфические виды файлов, например каталоги, ссылки, файлы устройств для доступа к компонентам аппаратного обеспечения. В команде ls -F такие специфические файлы обозначаются дополнительным символом.

```
user$ ls -lF
... 13. Apr 11:31 bak/
... 11. Apr 12:21 grepalltex*
...
```

В табл. 7.4 показана идентификация специальных файлов.

Таблица 7.4. Идентификация специальных файлов

Символ	Значение
/	Каталог
*	Исполняемый файл
@	Символьная ссылка
-	Символьное устройство
+	Блочное устройство
=	Pipe, FIFO

7.2. Ссылки

Ссылки — это указания на файлы. С помощью ссылок из различных участков структуры каталогов можно получать доступ к определенному файлу без необходимости сохранять несколько копий файла в различных местах жесткого диска. Таким образом, ссылки очень важны и помогают избежать избыточности в системе. В файловой системе Linux ссылки чаще всего встречаются в каталогах `/bin` и `/lib` (рекомендую внимательнее рассмотреть, например, `/usr/bin` или `/usr/lib` с `ls -l`).

Чтобы лучше понять, что такое ссылки, приведу пример. Допустим, в каталоге test находится файл `abc`. Логично, что команда `ln abc xyz` должна создавать новый файл `xyz`. Но на самом деле `abc` и `xyz` — это всего лишь две ссылки на один и тот же файл. Чтобы проверить, не имеем ли мы дело именно с такой ситуацией, необходимо воспользоваться командой `ls` с параметром `-l`. Во втором столбце вывода команды сообщается, сколько ссылок указывает на один и тот же файл (в данном примере — две). Если дополнительно используется параметр `-i`, то `ls` также называет индексный узел файла, идентичный у ссылок, указывающих на файл (и только у них).

```
user$ ls -li
 59293 -rw-r--r-- 1 root root 1004 Oct 416:40 abc
user$ ln abc xyz
user$ ls -li
 59293 -rw-r--r-- 2 root root 1004 Oct 416:40 abc
 59293 -rw-r--r-- 2 root root 1004 Oct 416:40 xyz
```

Если теперь вы измените любой из двух файлов (все равно какой), то автоматически изменится и другой файл (так как в действительности существует только один файл). Если удалить один из этих двух файлов, то просто уменьшится количество ссылок.

ПРИМЕЧАНИЕ

При обработке связанных ссылок в текстовом редакторе иногда могут получаться странные результаты: после первого сохранения ссылка указывает на резервный файл, а при повторном сохранении — в пустоту.

Причина такова: при сохранении некоторые редакторы создают файл резервной копии, переименовывая при этом имеющийся файл. Например: файл abc и копия abc~. Измененный файл сохраняется заново, получает новый индексный узел и поэтому не связан со ссылками. Чтобы с этим справиться, используйте символьные ссылки.

Символьные ссылки. В Linux различаются два вида ссылок. В предыдущем примере были показаны жесткие ссылки в таком виде, как они обычно создаются командой `ln`. Если же, напротив, применяется параметр `-s`, то команда создает символьные ссылки. Символьные ссылки имеют определенное достоинство по сравнению с жесткими ссылками — они могут указывать из файловой системы одного физического жесткого диска на файлы и каталоги другого диска. (Как правило, с жесткими ссылками невозможно ни то ни другое. Особую категорию образуют жесткие ссылки на каталоги. Их можно ставить, но эта возможность предоставляется только суперпользователям.)

Команда `ls` при работе с символьными ссылками позволяет указать, где находится исходный файл. В любом случае команда не предоставляет счетчик, с помощью которого можно было бы указать, из какого количества мест поставлены ссылки на исходный файл.

Внутреннее отличие между жесткими и символьными ссылками заключается в том, что в первом случае сохраняются данные об индексном узле, а во втором случае — имя файла или (если ставится ссылка на файл, расположенный за пределами каталога) путь к этому файлу.

```
user$ ln -s abc efg
user$ ls -li
 59293 -rw-r--r-- 2 root root 1004 Oct 416:40 abc
 59310 lrwxrwxrwx 1 root root 3 Oct 416:52 efg -> abc
 59293 -rw-r--r-- 2 root root 1004 Oct 416:40 xyz
```

СОВЕТ

Перед тем как создавать символьную ссылку, всегда необходимо перейти в тот каталог, в котором она будет содержаться, иначе ссылка может указать не туда, куда вы ожидали.

Символьные ссылки работают немного иначе, нежели жесткие. При удалении исходного файла (в том числе `abc`) ссылка на этот файл не изменяется, но теперь `efg` указывает в пустоту. Если же, напротив, удаляется символьная ссылка, то на исходный файл это никак не влияет.

Символьные ссылки могут указывать не только на файлы, но и на каталоги. В данном случае возможна некоторая путаница — одна символьная ссылка может вызвать эффект «дублирования» целого дерева каталогов. Однако на самом деле символьная ссылка представляет собой всего лишь дополнительный путь к тем же самым файлам и подкаталогам.

Как правило, при выставлении ссылок необходимо пользоваться только относительными указаниями путей и избегать абсолютных. Таким образом вы избавитесь от проблем, возникающих при подключении каталогов через NFS или при их перемещении.

Итак, определенные достоинства есть и у жестких, и у символьных ссылок. С символьными ссылками проще работать. Однако жесткие ссылки занимают меньше места на диске и работают быстрее.

7.3. Типы файлов (MIME)

Когда вы выбираете ссылку на MP3-файл в браузере или менеджере файлов, этот файл автоматически воспроизводится в проигрывателе. Если такая связь работает, это значит, что типы MIME в системе сконфигурированы правильно.

Аббревиатура MIME означает Multipurpose Internet Mail Extensions — «многоцелевые расширения электронной почты». Первоначально типы MIME разрабатывались для приложений к электронным письмам. Если к письму присоединялся, например, файл в формате PostScript или JPEG, то клиент электронной почты сразу знал, с помощью какой программы следует просмотреть или обработать этот файл. Чтобы данный механизм работал, требовалось настроить конфигурацию MIME.

Однако со временем область применения типов MIME существенно расширилась: когда вы переходите к файлу по ссылке, стоящей в браузере или в файловом менеджере, программа должна знать, как поступить с данным файлом. Итак, правильная конфигурация типов MIME важна в любых программах, которые должны уметь работать с файлами различных типов.

Конфигурация MIME

Linux не была бы Linux (или UNIX), если бы в системе имелось только центральное место для проведения конфигурации MIME. Такие места, конечно, есть, но они не централизованы, и их очень много. Данные MIME для программ KDE, программ Gnome, различных браузеров, системы печати CUPS и т. д. управляются по отдельности. Кроме того, есть и центральная система конфигурации MIME для всех тех программ, которые не располагают собственными файлами MIME.

Существуют разные причины того, что конфигурация MIME в Linux была распределена по нескольким местам. При работе KDE и Gnome применяется концепция, в соответствии с которой для обработки различных типов данных используются разные компоненты. Если в файловом менеджере программы KDE необходимо открыть файл-изображение в формате PNG, то в систему просто загружаются, а затем выполняются соответствующие компоненты. Поскольку библиотеки KDE и Gnome, как правило, несовместимы друг с другом, попытка файлового менеджера KDE выполнить компоненты Gnome (и наоборот) привела бы к катастрофическим последствиям. Во избежание такой ситуации KDE и Gnome используют собственные отдельные базы данных по MIME. По сходным причинам собственная конфигурация MIME присваивается и другим программам.

При работе со многими конфигурационными файлами MIME необходимо различать глобальную и индивидуальную конфигурацию, то есть общие настройки и настройки для отдельно взятого пользователя. Далее показана базовая конфигурация компонентов MIME в системе Linux. Пользовательские компоненты, связанные с MIME, описаны в других главах, например конфигурация типов MIME для KDE — в главе о KDE (см. главу 3) и т. п.

Общая конфигурация MIME. Общие файлы конфигурации MIME используются только теми программами, которые не располагают собственными файлами MIME. Настройки разделены на два файла, один из которых представляет собой глобальную версию, а другой — пользовательскую (табл. 7.5).

Таблица 7.5. Конфигурационные файлы MIME

Файл	Значение
/etc/mime.types	Глобальная конфигурация для типов файлов
/etc/mailcap	Глобальная конфигурация для программ
~/.mime.types	Локальная конфигурация для типов файлов
~/.mailcap	Локальная конфигурация для программ

Файл `mime.types` содержит список, в котором представлено соответствие между типами файлов (первый столбец) и расширениями файлов (все остальные столбцы). Например, типу `application/pdf` соответствует расширение PDF. В списке

mime.types различаются текстовые приложения и X-приложения, типы MIME которых имеют вид application/x-name.

```
# в /etc/mime.types
...
application/pdf    pdf
```

Файл mailcap указывает, какая программа должна применяться для отображения или обработки файла того или иного типа. В следующей строке сообщается, что для отображения файлов PDF следует использовать программу evince. В отличие от mime.types, столбцы mailcap должны разделяться точками с запятой; %s — это подстановочный символ для имен файлов.

```
# в /etc/mailcap
application/pdf; evince %s
```

Магические файлы для распознавания типа файла

Тип MIME предназначен для того, чтобы те или иные файлы открывались или обрабатывались программами, предназначенными специально для этого. Но как вообще удается установить тип обрабатываемого файла? Как правило, тип файла определяется по его расширению. Например, расширение *.ps означает, что перед нами файл PostScript.

Если файл не имеет такого идентификатора, программа file или соответствующие программы систем KDE или Gnome пытаются вычленить эквивалент типа файла по первому байту, содержащемуся в этом файле, либо по характерным для него последовательностям символов. Метод распознавания основывается на информации о том, какие байты и типичные последовательности символов может содержать файл; эта информация встраивается в команду file при компиляции. В некоторых дистрибутивах стандартную конфигурацию можно изменять с помощью файлов /etc/magic или ~/.magic.

7.4. Поиск файлов (find, grep, locate)

В системе Linux предусмотрено множество возможностей поиска файлов (табл. 7.6). Выбор команды, лучше всего подходящей для конкретного случая, зависит от того, о файле какого типа идет речь (текстовый файл, программа и т. д.) и какая информация о файле известна (фрагменты названия, фрагменты содержимого и т. д.).

Таблица 7.6. Инструменты для поиска файлов

Команда	Функция
grep	Поиск текста в текстовом файле
find	Поиск файлов по имени, дате, размеру и т. д.
locate	Поиск файлов по имени
whereis	Поиск файлов в заданных каталогах
which	Поиск программ в PATH-каталогах

Команды which и whereis

Команда which ищет указанную команду. Она возвращает полное имя команды, которая была бы выполнена, если бы вы набрали ее имя без указания пути. Команда which осуществляет поиск только по каталогам, указанным в PATH, работая при этом исключительно быстро. PATH содержит список каталогов, в которых находятся программы. Однако необходимо учитывать, что PATH для администратора содержит больше каталогов, чем для обычного пользователя. Итак, если вы ищете системные команды, необходимо войти в систему с правами администратора.

```
user$ which emacs
/usr/bin/emacs
```

Команда whereis просматривает все пути, обычно используемые для двоичных файлов, файлов конфигурации, man-страниц и исходного кода. Для этого ее нужно ввести после имени файла. Команда whereis охватывает больше файлов, чем which, при этом поиск не ограничивается одними только программами. Она не может искать в каталогах, которые специально не указаны программе как места для поиска (см. раздел справки man whereis).

```
user$ whereis fstab
fstab: /etc/fstab /usr/include/fstab.h /usr/share/man/man5/fstab.5.gz
```

Команда locate

Команда locate ищет файлы, в полном названии которых (путь + имя файла) содержится указанная схема поиска. Поиск осуществляется очень быстро: locate не ищет по файловой системе, а обращается к базе данных, в которой содержится список имен всех файлов из файловой системы. В зависимости от дистрибутива locate показывает все файлы, к которым пользователь может получить доступ. Если вы ищете системные файлы, войдите в систему как администратор и выполните команду locate. Использовать ее можно только в том случае, когда в системе установлен соответствующий пакет, который по умолчанию имеется не во всех дистрибутивах.

Примеры. Следующая команда ищет конфигурационный X-файл xorg.conf:

```
user$ locate xorg.conf
/etc/X11/xorg.conf
/etc/X11/xorg.conf.backup
/etc/X11/xorg.conf~
/usr/share/man/man5/xorg.conf.5x.gz
```

Поиск dvips (возможен, если установлен и этот пакет, и LATEX) выдает много результатов, так как результат поискового запроса имеется во многих названиях каталогов. Все эти результаты не отображаются, а подсчитываются с помощью параметра wc.

```
user$ locate dvips | wc -l    421
```

Количество результатов значительно уменьшится, если искать только те файлы, названия которых оканчиваются на dvips:

```
user$ locate ,*dvips'
/usr/bin/dvips
/usr/bin/odvips
/usr/bin/opdvips
/usr/bin/pdvips
/usr/local/texmf/dvips
/usr/local/texmf/fonts/map/dvips
...
```

Команда updatedb. Качество результатов поиска зависит от актуальности базы данных, используемой командой `locate`. В большинстве дистрибутивов эта база ежедневно обновляется с помощью команды `updatedb`. Разумеется, `updatedb` можно выполнять вручную всякий раз, как это потребуется. Для этого необходимо обладать правами администратора.

Детали, характерные для определенных дистрибутивов. Внедрение команд `locate` и `updatedb` отличается от дистрибутива к дистрибутиву. В Debian, Fedora и Ubuntu эти команды входят в состав стандартного пакета mlocate. Файловая база данных находится в файле `/var/lib/mlocate/mlocate.db` и ежедневно обновляется программой Cron-Job `/etc/cron.daily/mlocate`. Конфигурационный файл `/etc/updatedb.conf` определяет, какие каталоги и файловые системы не будут учитываться при поиске (например, CD, DVD, различные буферные каталоги).

В стандартной сборке openSUSE команда `locate` отсутствует. Прежде чем вы сможете пользоваться этой поисковой командой, потребуется установить пакет `findutils-locate` и один раз выполнить `updatedb`, находясь в системе с правами администратора. В дальнейшем эта команда будет ежедневно выполняться программой Cron-Job `/etc/cron.daily/suse.de-updatedb`. Конфигурация осуществляется через `/etc/sysconfig/locate`.

Команды find и grep

Команда `find` очень мощная, но не менее сложная, чем описанные выше команды для поиска файлов. Она учитывает различные поисковые критерии (схемы поиска по именам файлов, размеру файлов, датам создания файла или последнего обращения к нему и т. д.). Полная справка по параметрам этой команды содержится в `man find`. Однако лучше всего будет продемонстрировать использование `find` на примерах, приведенных ниже. Учитывайте, что `find` работает сравнительно медленно, так как она просматривает всю файловую систему каталог за каталогом.

Команда find

Если не указывать дополнительные параметры, `find` выдает список всех файлов, находящихся в текущем каталоге и всех его подкаталогах:

```
user$ find
...
```

Следующая команда ищет все файлы в текущем каталоге и во всех подкаталогах, названия которых начинаются с `.e`:

```
user$ find -name '.e*'
./.evolution
./.emacs
./.emacs~
./.esd_auth
...
```

Команда find производит поиск, начиная с каталога /usr/share/texmf, и ищет все файлы вида *.tex в каталоге, название которого оканчивается на latex.

```
user$ find /usr/share/texmf -path '*latex/*.tex'
/usr/share/texmf/ptex/platex/base/plnews03.tex
/usr/share/texmf/ptex/platex/base/kinsoku.tex
...
```

В следующем примере find ищет все каталоги, находящиеся в /etc/. Обычные файлы, располагающиеся в /etc/., среди результатов не показываются. Список результатов упорядочивается по алфавиту с помощью команды sort (по умолчанию такой сортировки не происходит).

```
root# find /etc -type d | sort
/etc
/etc/acpi
/etc/acpi/actions
...
```

В следующем примере find ищет все файлы в (под)каталогах /home, принадлежащих пользователям группы users, причем искомые файлы должны были каким-либо образом быть изменены в течение последних пяти дней (содержание, права доступа и т. д.):

```
root# find /home -group users -mtime -5
...
```

Команда find -mtime +5 находит такие файлы, которые были изменены *ранее*, чем пять дней назад, а команда -mtime 5 возвращает файлы, которые были изменены *ровно* пять дней назад. При этом во многих случаях «вчера» означает для find «24 и более часов от данного момента». Если вместо -mtime применить параметр -ctime, то точкой изменения времени будет считаться «время изменения индексного дескриптора» (inode change time). Эта точка изменяется и тогда, когда изменились лишь права доступа, а содержимое каталога осталось прежним.

Следующая команда удаляет все резервные копии, содержащиеся в данном каталоге и во всех подкаталогах. При этом find строит список всех сомнительных файлов и передает его команде rm через подстановку команды ($(*команда*)).

```
user$ rm $(find . -name '*~')
```

Если речь идет об *очень* большом количестве файлов, то при выполнении вышеуказанной команды возникает ошибка: запись со всеми *~-файлами может получиться настолько длинной, что превысит размер командной строки. В таких случаях следует воспользоваться либо параметром -exec- команды find, либо командой xargs.

Команда grep

Эта команда ищет в текстовом файле соответствия для заданной поисковой схемы. В зависимости от того, какие параметры настроены в конкретном случае, команда может дополнительно показывать найденные фрагменты текста или просто сообщать, в каком количестве строк была найдена заданная поисковая схема. Поисковая схема является так называемым регулярным выражением.

Следующая команда просматривает все TEX-файлы текущего каталога в поисках последовательности символов `emacs`. Список всех найденных строк (перед каждой из которых указывается имя файла) отображается на экране.

```
user$ grep emacs *.tex
...
```

Команда `grep` определяет, как часто применяется функция `arctan` в указанных C-файлах.

```
user$ grep -c arctan\(.*\) *.c
```

Команда `grep` с параметром `-v` возвращает в качестве результата все строки, в которых отсутствует заданный шаблон поиска.

В приведенном далее примере `grep` удаляет из `configfile` все строки, которые начинаются с символа # (то есть все комментарии). Следующая команда `cat` дополнительно удаляет все пустые строки. Конечный результат сохраняется в файле `nocomments`.

```
user$ grep -v '^#' configfile | cat -s > nocomments
```

Эта команда очень удобна, когда всего несколько строк конфигурационного кода приходится на сотни или тысячи строк комментариев.

Комбинирование find и grep

Команды `find` и `grep` можно комбинировать, чтобы выполнять расширенный поиск. В следующем примере команда `find` просматривает все файлы на предмет того, нет ли в них последовательности символов `emacs`. Если такая последовательность обнаруживается, то название файла выводится на экран. Обратите внимание на то, что нельзя указывать параметр `-print` перед `-exec` (в отличие от предыдущего примера, команда `grep emacs *.tex` учитывает все файлы с расширением TEX независимо от глубины вложения подкаталога, в котором они могут находиться).

```
user$ find -name '*.tex' -type f -exec grep -q emacs {} \; -print
...
```

Следующая команда просматривает в текущем каталоге все файлы размером менее 10 Кбайт на предмет наличия в них регулярного выражения `case.*in`. Список найденных файлов сохраняется в файле `result`. Ограничение размера файла вводится для того, чтобы исключить из поиска двоичные файлы (обычно они гораздо больше 10 Кбайт).

```
user$ find -name '*' -maxdepth 1 -size -10k -exec grep -q \
   > case.*in {} \; -print > result
```

7.5. Запись CD и DVD

О важнейших программах с пользовательским интерфейсом — Brazero и K3B — я уже рассказал в главах о Gnome и KDE. Если вы записываете диски лишь иногда, от случая к случаю, то используйте эти программы — в них вы найдете все необходимые функции, к тому же с ними легко работать. Если же вы ищете пользовательскую программу для работы с текстовой консолью, то вам очень понравится команда `burncdda`.

В этом разделе будут рассмотрены команды, которые выполняются на фоне основной программы. Они могут быть интересны, например, в тех случаях, когда вам нужен автоматизированный сценарий для записи CD с резервными копиями. Как и во многих других случаях, понимание принципа работы этих команд поможет вам лучше понимать саму концепцию работы Linux.

Названия устройств. Прежде чем начать работу с командами, описанными ниже, нужно знать название устройства; по такому названию вы будете обращаться к приводу. Как правило, верное название устройства таково: `/dev/scd0`, `/dev/scd1` и т. д. или `/dev/sr0`, `/dev/sr1` и т. д.

Для некоторых команд название устройства необходимо указывать в виде тройки чисел (например, `dev=3,0,0`). В эту тройку входят: номер шины SCSI (обычно `0`), SCSI-ID устройства и, наконец, *логический номер устройства* (коротко ЛНУ, обычно также `0`). Верную комбинацию чисел для вашего привода вам проще всего будет узнать с помощью команды `readcd -scanbus`.

СОВЕТ

Независимо от того, какой именно диск вы записываете — CD или DVD, убедитесь, что дистрибутив Linux не встраивает носитель с данными в дерево каталогов, и просто отключите этот носитель от дерева каталогов!

Создание и тестирование ISO-образов

Прежде чем вы сможете записать данные на CD или DVD, вам потребуется так называемый *образ диска* (ISO). Это файл, содержащий другие файлы, подготовленные для записи во внутреннем формате оптического носителя данных. Как правило, для создания образов дисков используется команда `genisoimage` (ранее — `mkisofs`).

Команда genisoimage. Команда `genisoimage` позволяет записать на диск содержимое одного или нескольких каталогов. Формат ISO-9660, предусмотренный для CD, оперирует собственным очень ограниченным набором символов, в котором допускаются лишь немногие символы, кроме тех, что относятся к кодировке ASCII. Чтобы справиться с этим недостатком, существует множество расширений стандарта ISO, наиболее распространенными из которых являются следующие два. Оба они поддерживаются `genisoimage`.

- Расширение Rockridge, обычное в системах UNIX/Linux, обеспечивает сохранение длинных названий файлов в форме любых, нуль-терминированных последовательностей символов. Кроме того, это расширение позволяет сохранять права доступа (UID, GID, биты доступа).

В любом случае диски, записанные с помощью Rockridge, не содержат информации о том, в какой кодировке создавался образ диска. Это может вызывать проблемы, если позже носитель данных будет использоваться на компьютере, который поддерживает другую кодировку. Проще всего объяснить это на примере: пару лет назад в Linux еще широко применялась кодировка Latin-1. Диски с данными, создававшиеся тогда с помощью расширения Rockridge, записывались именно в этой кодировке. Если сегодня вы используете такие диски на одном из современных дистрибутивов, в которых применяется кодировка Unicode (UTF-8), то символы, которые встречаются в названиях файлов и не относятся к ASCII, будут интерпретированы неправильно. Если уже перед подготовкой ISO-образа известно, что создаваемый диск будет применяться на компьютере с другой кодировкой, то можно настроить нужную кодировку с помощью параметра `-output-charset`.

- Расширение Joliet, которое часто используется и в системах с Windows, также позволяет сохранять длинные названия файлов, при этом применяется кодировка Unicode (UTF-16).

Примеры. Следующая команда записывает все файлы, содержащиеся в каталоге `/master`, в файл `/tmp/master.iso`. Сам каталог master *не* хранится в образе диска. Образ диска использует расширение Rockridge (параметр `-r`), а также расширение Joliet (параметр `-J`) и получает название `Linux` (параметр `-V`). Если вы записываете ISO-образ на CD, то данная последовательность символов становится именем диска.

```
user$ genisoimage -o /tmp/master.iso -r -J -V Linux /master
```

Второй пример напоминает первый, но теперь мы создаем загрузочный диск:

```
user$ genisoimage -o /tmp/master.iso -r -J /master -b images/boot.img -c boot.catalog
```

В третьем примере каталог master сам становится каталогом в образе диска (параметр `-graft-points`):

```
user$ genisoimage -o /tmp/master.iso -r -graft-points /master=/master
```

СОВЕТ

Если вы не работаете с параметром –r, следите за тем, чтобы все файлы каталога master относились к корневому каталогу и прочитывались!

```
user$ chown -R root.root /master
user$ chmod -R a+r /master
```

Команда xorriso. В качестве альтернативы для команды `genisoimage` можно применять команду `xorriso` из одноименного пакета. С помощью этой команды можно создавать файлы образов ISO, изменять их и записывать на CD и DVD. Массу примеров применения этой программы вы найдете на сайте, посвященном ее разработке: http://www.gnu.org/software/xorriso/.

Команда dd. Если вы хотите скопировать без изменений CD или DVD с данными (но не аудиодиск!), вам хватит одной команды `dd`, чтобы создать необходимый для этого файл образа диска. Вместо `/dev/cdrom` укажите имя устройства вашего CD- или DVD-привода, которое отличается в зависимости от дистрибутива.

```
user$ dd if=/dev/cdrom of=/usr/local/iso.img bs=2048
```

Команда readcd. Альтернативой dd является команда readcd или readom. При этом для считывания информации с CD используются команды SCSI, это дает те же результаты, что и применение dd. Я проводил испытания на двух разных компьютерах, и в обоих случаях readcd выдавала бесчисленные сообщения об ошибках. В отличие от работы с dd, требовалось указать привод CD или DVD с помощью тройки чисел. Правильную комбинацию чисел для вашего привода можно узнать с помощью команды readcd -scanbus.

```
user$ readcd dev=0,0,0 f=iso.img
```

Достоинство readcd заключается в том, что она также может создавать TOC-файл (при работе с аудиодисками — с параметром -clone), и в том, что программа, в зависимости от указанного параметра, будет по-разному обращаться с ошибками считывания (параметры -noerror и -noclone). С помощью параметра -w можно применять readcd и для записи CD.

Тестирование ISO-образа. С помощью так называемого *петлевого устройства* ядра Linux можно рассмотреть любой файл как файловую систему и, используя команду mount, связать его с деревом каталогов. Функция петлевого устройства находится в модуле ядра loop, который входит в состав всех наиболее распространенных дистрибутивов. Если этот модуль не загружается автоматически, попробуйте вариант с modprobe. Следующая команда связывает файловую систему образа диска, содержащуюся в файле master.iso, с деревом каталогов в режиме «только для чтения»:

```
root# mkdir /iso-test
root# mount -t iso9660 -o loop,ro /tmp/master.iso /iso-test/
```

Теперь можно просмотреть в каталоге iso-test содержимое будущего CD.

Запись CD

Программа cdrecord и команда wodim. Уже более десяти лет команда cdrecord применяется для записи CD. С лета 2006 года разработчик cdrecord Йорг Шиллинг использует при работе с некоторыми составляющими одноименного пакета лицензию компании Sun под названием CDDL. Другие разработчики считают, что эта лицензия несовместима с GPL. Именно поэтому случилось так называемое ветвление, то есть раскол проекта: последняя версия cdrecord, соответствующая GPL, послужила основой для команды wodim (write data to optical disk media)[1], относящейся к новому проекту cdrkit. Параллельно с cdrecord существует еще два ветвления, две новые команды: из mkisofs получилась genisoimage, а из cdda2wav — icedax.

Во всех нынешних дистрибутивах применяются команды wodim, genisoimage и icedax. Однако из соображений совместимости cdrecord можно вызывать по старому названию cdrecord; /usr/bin/cdrecord — это только ссылка на wodim.

Перед тем как записать компакт-диск с данными, вам, как правило, потребуется записать ISO-образ с помощью genisoimage. Следующие команды сначала

[1] Запись данных на оптические диски.

имитируют запись CD с данными (-dummy), а потом осуществляют его на самом деле:

```
root# wodim -dummy -v speed=16 dev=/dev/scd0 iso.img
root# wodim -v speed=16 dev=/dev/scd0 iso.img
```

Если компьютер достаточно быстрый, вы можете связать genisoimage и wodim с помощью символа вертикальной линии. Так можно сэкономить место для ISO-образа:

```
root# genisoimage -r /master | wodim -v speed=16 dev=/dev/scd0 -
```

Следующая команда создает аудиодиск. Исходными данными являются файлы в формате WAW. Они обрабатываются в алфавитном порядке. Если вы хотите задать другой порядок, нужно перечислить файлы желаемым образом.

```
root# wodim -v speed=16 dev=0,5,0 -pad -dao -audio *.wav
```

Команда cdrdao. Это альтернатива для wodim. Возможности cdrdao не так широки, но она предлагает гораздо больше параметров для считывания и записи аудиодисков. Название команды подсказывает, что информация создается в режиме односеансовой записи (disk at once, коротко — DAO).

На практике команда cdrdao чаще всего используется для копирования аудиодисков. Первая команда cdrdao создает файлы data.bin (содержимое компакт-диска) и data.toc (оглавление). Вторая команда записывает эти данные на CD.

```
user$ cdrdao read-cd --device 0,0,0 data.toc
user$ cdrdao write --device 0,0,0 --buffers 64 data.toc
```

Верификация дисков с данными. Следующая команда сравнивает содержимое CD с оглавлением каталога master файл за файлом и байт за байтом. Все найденные различия записываются в файл diff.log, находящийся в домашнем каталоге. Вместо /media/cdrom необходимо указать каталог в вашей файловой системе, к которому будет привязан CD.

```
root# diff -qrd /master /media/cdrom/ >& ~/diff.log
```

Во втором окне (или во второй консоли) можно проследить процесс создания файла diff.log с помощью tail. Поскольку в данном случае используются символьные ссылки, не обойдется без сообщений об ошибках, так как символьные ссылки уже не будут указывать в нужное место CD. Однако по-настоящему беспокоиться стоит в том случае, когда отдельные файлы вообще не удается прочитать (ошибка ввода-вывода) или когда оглавление файлов отличается от необходимого (а вы уверены, что не вносили в файл изменений).

```
root# tail -f ~/diff.log
```

Если вы хотите просто протестировать, можно ли прочитать все данные содержащиеся на диске (вне зависимости от того, что это за данные), выполните следующую команду. Такой тест оправдан, например, в тех случаях, когда вы получили компакт-диск, который, возможно, неисправен (например, это касается установочного диска Linux).

```
root# dd if=/dev/cdrom of=/dev/null
```

Запись DVD

При записи DVD вам на выбор предлагается целый ряд команд или пакетов.

- Самая популярная команда — `dvd+rw-tools`, которую мы кратко рассмотрим в данном подразделе.
- Если с этой командой возникнут проблемы, можно попытать счастья с `wodim`. Она подходит для записи обычных DVD-R и DVD+R, причем ее синтаксис не отличается от применяемого при записи CD. В любом случае `wodim` предлагает меньше параметров для записи на носители DVD+RW или DVD-RW.

Команда dvd+rw-tools

Все команды, показанные далее в этом подразделе, входят в состав пакета `dvd+rw-tools`. Первоначально он обеспечивал только поддержку форматов DVD+R и DVD+RW (отсюда и название). Однако теперь с его помощью можно записывать также DVD-R и DVD-RW, а также Blu-ray-диски (последний вариант я не пробовал). Пакет `dvd+rw-tool` по умолчанию установлен во всех дистрибутивах, распространенных в настоящее время. Более подробная информация находится по адресу http://fy.chalmers.se/~appro/linux/DVD+RW/.

Команда growisofs

Это основная команда пакета `dvd+rw-tools`. Она записывает диски DVD+R, DVD+RW, DVD-R, DVD-RW и Blu-ray. Рассмотрим общую информацию по разным типам носителей.

- DVD+R, DVD-R — данные можно записать на диск как при мультисессии. При первой сессии используется команда `growisofs -Z`, при всех остальных сессиях — `growisofs -M`. Удалить записанные данные уже нельзя. Форматирование диска произвести невозможно.
- DVD+RW, DVD-RW — перед первым применением носитель необходимо отформатировать с помощью команды `dvd+rw-format`. Кроме того, можно добавлять данные на DVD+R/DVD-R в несколько этапов. Если вы хотите записать данные на место информации, уже содержащейся на диске, то просто запустите новый цикл сессий с помощью команды `growisofs-Z`. В отличие от записи CD-RW, заново форматировать DVD в данном случае не требуется.

При записи DVD-RW в зависимости от форматирования поддерживаются режимы *последовательных добавлений* и *ограниченной перезаписи*. Поскольку при работе этой команды используется команда `genisoimage`, большинство параметров этих команд идентичны. Следующая команда сохраняет содержимое каталога data на DVD. Параметры команды `genisoimage -r` и `-J` влияют на то, что DVD получает длинные названия, необходимые для работы с расширениями Rockridge и Joliet. Вместо имени устройства `/dev/sr`*n* нужно в зависимости от дистрибутива указывать `/dev/scd`*n*.

```
user$ growisofs -r -J -Z /dev/sr0 data/
```

Вторая сессия производится следующим образом (параметр `-M` вместо `-Z`):

```
user$ growisofs -r -J -M /dev/sr0 moredata/
```

ПРИМЕЧАНИЕ

Обратите внимание, что перед началом новой сессии необходимо извлечь DVD, а потом вновь поместить его в привод!

Чтение мультисессионных DVD может протекать с проблемами на некоторых приводах. При работе с DVD-RW следует использовать режим ограниченной перезаписи.

Некоторые DVD-приводы вообще не могут работать с DVD+RW (как с моносессионными, так и с мультисессионными). С помощью команд, указанных ниже, эти проблемы иногда удается устранить. Команда dvd+rw-format записывает на DVD+RW область для вывода (DVD не форматируется, все данные сохраняются!), а dvd+rw-booktype изменяет на диске информацию о типе (Book-type).

```
user$ dvd+rw-format -lead-out /dev/sr0
user$ dvd+rw-booktype -dvd-rom -media /dev/sr0
```

Обычно команда `growisofs` сообщает все параметры, кроме `-Z` или `-M`, команде `genisoimage`, а затем записывает результат выполнения `genisoimage` прямо на DVD. Если вы хотите записать уже имеющийся образ диска (ISO), синтаксис будет такой: `-Z device=isofile`:

```
user$ growisofs -Z /dev/sr0=data.iso
```

Формат dvd+rw-format

DVD+RW и DVD-RW перед первым применением необходимо отформатировать (когда использовать режим ограниченной перезаписи, рассказано ниже). Эту функцию выполняет команда `dvd+rw-format`:

```
user$ dvd+rw-format /dev/sr0
```

Детали процесса форматирования немного отличаются в зависимости от типа носителя.

- DVD+RW — в данном случае форматируется только начальная область болванки. Ее размер определяется в зависимости от записывающего устройства. Поэтому когда процесс форматирования завершается на отметке около 11,5 (или любой другой отметке меньше 100) — это не ошибка! Форматирование за пределами начальной области автоматически выполняется приводом, как только информация, записываемая на DVD, заполняет всю начальную область (отформатированную предварительно).
- DVD-RW — по умолчанию такие диски форматируются командой `dvd+rw-format` для ограниченной перезаписи. В этом режиме можно перезаписать на диск новую информацию на место уже имеющейся. При этом не требуется форматировать DVD-RW перед каждой новой сессией записи.

Если DVD-RW имеет параметр `-blank`, его можно форматировать и для записи в режиме последовательных добавлений. Этот режим особенно хорош для записи VideoDVD и улучшает совместимость с некоторыми программами-плеерами. В таком режиме команда `growisofs` не может перезаписывать данные. Для этого DVD каждый раз нужно заново форматировать — очень длительный процесс.

Итак, для оптимального взаимодействия с `growisofs` обязательно необходимо форматировать DVD-RW с помощью команды `dvd+rw-format` и без параметра `-blank`!

При форматировании не происходит физического удаления данных. Если вам требуется именно физическое удаление, например из соображений информационной

безопасности, лучше выполнить команду вида `-Z device=/dev/zero`. Таким образом информация на носителе стирается полностью.

Команда dvd+rw-mediainfo

Если у вас есть диск и вы не знаете, какого он типа, заполнен ли уже этот диск информацией, и если да, то в каком режиме и с каким количеством сессий, эту информацию можно узнать с помощью команды `dvd+rw-mediainfo`:

```
user$ dvd+rw-mediainfo /dev/sr0
INQUIRY:                [_NEC ][DVD_RW ND-1300A ][1.07]
GET [CURRENT] CONFIGURATION:
 Mounted Media:         1Ah, DVD+RW
GET PERFORMANCE:
 Speed Descriptor#0:    00/221280 Reading@7.8x Writing@2.3x
READ DVD STRUCTURE[#0h]:
 Media Book Type:       92h, DVD+RW book [revision 2]
 Media ID:              RICOHJPN/W01
 Legacy lead-out at:    221280*2KB=453181440
...
```

7.6. Права доступа, пользователи и принадлежность к группам

Linux изначально создавалась как многопользовательская система, поэтому ей требуются механизмы, управляющие тем, кто и к каким данным имеет доступ, кто может изменять такие права и т. д. Основу системы доступа образуют принципы управления пользователями и группами, описанные в разделе 11.4.

Права доступа к файлу

О каждом файле или каталоге сохраняется следующая информация:

- владелец файла;
- группа, к которой относится файл;
- девять битов доступа (`rwxrwxrwx` для `read/write/execute` — для владельца файла, членов группы владельца и всех остальных);
- еще несколько дополнительных битов для выполнения специальных функций.

Как правило, владельцем файла является его создатель. В качестве группы обычно используется основная группа владельца.

Информация о доступе (`r`, `w` и `x`) описывает, кто имеет право читать файл, записывать в него информацию (то есть вносить изменения) и выполнять его. Таким образом, владелец файла получает больше прав, чем другие пользователи. Обычно эта информация называется битами доступа, так как внутри системы она сохраняется как число с разрядной кодировкой.

Биты доступа, владелец файла, а также отнесенность этого файла к той или иной группе можно просмотреть с помощью команды `ls -l`. Для обычного текстового файла `ls` возвращает примерно такой результат:

```
michael$ ls -l файл.txt
-rw-r--r--   1 michael   users    3529 Oct   415:43 файл.txt
```

Коротко объясню эти данные: первый символ указывает тип файла (в данном случае для обычного файла, для каталога ставится `d` (directory), `l` — для символьной ссылки (link) и т. д.). Поскольку мы имеем дело с текстовым файлом, первый `x`-бит деактивирован, то есть файл нельзя выполнять. Все остальные пользователи, независимо от того, относятся ли они к группе users, могут читать этот файл, но не могут его изменять.

Если michael пожелает, чтобы этот файл был доступен для чтения только пользователям группы users и был недоступен для пользователей, не входящих в эту группу, нужно будет деактивировать последний `r`-бит. Для этого применяется команда `chmod`.

```
michael$ chmod o-r файл.txt
michael$ ls файл.txt -l
-rw-r-----   1 michael  users    3529 Oct    415:43 файл.txt
```

Допустим, требуется предоставить доступ к чтению файла *файл*`.txt` только двум пользователям: michael и kathrin. Для этого можно создать новую группу, к которой будут относиться оба этих пользователя. Если michael и kathrin вдвоем работают в группе по разработке документации в определенной фирме, можно назвать эту группу, например dokuteam. Дополнительно изменяем групповую отнесенность с помощью команды `chgrp`:

```
michael$ chgrp dokuteam файл.txt
michael$ ls файл.txt -l
-rw-r-----   1 michael   dokuteam   3529 Oct   415:43 файл.txt
```

Восьмеричное представление

Можно не использовать запись `rwxrwxrwx`, а представить восемь битов доступа и еще три специальных бита в восьмеричном виде: биты доступа для пользователей, группы и всех остальных пользователей соответственно представляются в виде одной цифры. Каждая цифра составляется из величин `4`, `2` и `1` для `r`, `w` и `x` соответственно. Таким образом, `660` означает `rw-rw----`, `777` означает `rwxrwxrwx`. Три специальных бита `Setuid`, `Setgid` и `Sticky` имеют восьмеричные значения `4000`, `2000` и `1000`.

С помощью команды `chmod` можно указывать биты доступа в восьмеричной форме, опытные пользователи предпочитают ее применять, чтобы не печатать лишнего:

```
user$ chmod 640 файл.txt
```

Поразительно, но команда `ls` не может представлять биты доступа в восьмеричном виде. Помочь в данном случае может следующая команда (к сожалению, абсолютно нечитабельная):

```
user$ ls -l | awk '{k=0;
                    for(i=0;i<=8;i++)k+=((substr($1,i+2,1)~/[rwx]/)*2^(8-i));
                    if(k)printf("%0o ",k);print}'
```

```
755 drwxr-xr-x 17 kofler kofler 4096 2008-10-28 15:34 php53-example
550 dr-xr-x---  2 kofler kofler 4096 2008-10-17 10:33 Private
755 drwxr-xr-x 10 kofler kofler 4096 2007-10-10 18:17 samples
```

Права доступа к каталогам

В принципе девять битов доступа применимы и при работе с каталогами, но в таком случае их значение несколько отличается: `r`-бит позволяет другим пользователям просмотреть содержимое каталога с помощью команды `ls`. Кроме того, `x`-бит дает возможность перейти в этот каталог с помощью `cd`. Если поставить и `x`, и `w`, то в каталоге можно создавать новые файлы.

Права доступа к устройствам

Доступ к различным аппаратным компонентам, например жестким дискам, приводам CD и DVD, интерфейсам и т. д., осуществляется в Linux через так называемые файлы устройств. Чтобы иметь возможность управлять тем, какой пользователь может обращаться к каким устройствам, они распределяются между различными группами пользователей. Например, устройства `/dev/ttyS*` предназначены для работы с серийными устройствами в системе Ubuntu в группе `dialout`:

```
root# ls -l /dev/ttyS1
crw-rw----    1 root    dialout    5, 65 Jul 18   /dev/ttyS1
```

Если системному администратору понадобится предоставить пользователю hubert права работы с серийным интерфейсом, то hubert будет добавлен в группу dialout:

```
root# usermod -a -G dialout hubert
```

Специальные биты

Понять значение трех троек битов доступа `rwxrwxrwx` совсем не сложно. С их помощью можно сохранить дополнительную информацию о файлах и каталогах. Как правило, об этих специальных битах должны знать только системные администраторы.

Бит Setuid

Бит `Setuid` иногда сокращенно называется `suid`. С его помощью программа всегда выполняется так, как если бы пользователь запускал ее сам. Итак, если владельцем программы является администратор (это случается довольно часто), то любой пользователь может работать с этой программой так же, как если бы он был администратором. Внутри системы для выполнения программы применяется пользовательский идентификационный номер владельца файла, а не UID пользователя, работающего с файлом в данный момент.

Бит `Setuid` применяется для того, чтобы присваивать обычным пользователям дополнительные права, действующие лишь при выполнении данной конкретной

программы. Правда, это чревато угрозой для безопасности — особенно тогда, когда при выполнении данной программы запускаются другие программы. Это означает, что по возможности следует избегать применения бита Setuid. Одно из немногих исключений — команда mount. Вместо бита Setuid можно использовать команду sudo.

В таких программах команда ls -l показывает для пользовательского бита доступа букву s (а не x). Восьмеричное значение этого бита (для chmod) составляет 4000.

```
root# ls -l /bin/mount
-rwsr-xr-x   1 root   root   68508 Feb 25   01:11   /bin/mount
```

Бит Setgit

Этот бит работает с программами почти так же, как и Setuid. Правда, в данном случае при выполнении программы используется идентификационный номер группы, к которой относится файл (а не групповой идентификационный номер пользователя, работающего с программой). В таких программах команда ls -l выдает для групповых битов доступа букву s (а не x). Восьмеричное значение этого бита составляет 2000.

При работе с каталогами бит Setgid действует так, что новые файлы, создаваемые в том или ином каталоге, сразу же относятся к той же группе, что и этот каталог (а не к группе пользователя, создавшего файл, как это бывает обычно).

Бит Sticky

В каталогах, в которых файлы может изменять любой пользователь, бит Sticky гарантирует, что каждый пользователь может удалять только свои файлы, но не файлы коллег. Этот бит ставится, например, для каталога /tmp. Здесь любой пользователь может сохранять временные файлы. Однако необходимо избегать таких случаев, в которых любой пользователь мог бы по собственному усмотрению переименовывать или удалять файлы других пользователей.

Команда ls -l в таких программах выдает для всех действительных битов доступа букву t (а не x). Восьмеричное значение этого бита составляет 1000. Не забывайте при этом, что значение бита Sticky является специфичным для Linux. В других системах UNIX значение этого бита может быть другим (или может вообще отсутствовать).

```
user$ ls -ld /tmp/
drwxrwxrwt   18 root   root   4096 Jun 14   15:34   /tmp/
```

Специальные биты в ls

При выполнении команды ls -l в определенной ситуации могут отображаться специальные биты S и T. При этом мы не имеем дело с новыми специальными битами, а получаем указание на то, что биты Setuid, Setgid или Sticky были использованы неправильно.

- S — бит Setuid или Setgid поставлен, но не поставлен бит доступа x (при этом Setuid или Setgid не будут работать).
- T — бит Sticky поставлен, но не поставлен бит доступа x для группы others.

Функция или тип файла

В системе Linux вместе с битами доступа и специальными битами также сохраняется информация о том, какую функцию выполняет файл. Например, это может быть обычный файл, каталог, ссылка, блочное устройство и т. д.

Большинство файловых менеджеров скрывают эту дополнительную информацию. Однако существуют и такие программы, которые отображают эти данные в виде числовых значений. Тогда полная спецификация (указание) для обычного файла равняется `10000` плюс *x*`000` (специальные биты) плюс *xxx* (биты доступа), то есть, к примеру, `100760`. Числовые коды, соответствующие файлам различных типов, можно узнать с помощью команды `man 2 stat`.

Владелец, группа и биты доступа для новых файлов

В этом подразделе будет рассмотрен вопрос о том, какие факторы определяют информацию о доступе к новым файлам. Чтобы просто опробовать этот механизм, воспользуйтесь командой `touch`. Она создает новый пустой файл, если указанный файл не существует.

Пример. Пользователь michael создает новый файл `myFile1`. Не удивляйтесь, что этот файл опять же принадлежит пользователю michael, ведь именно michael его создал. В качестве индикатора групповой принадлежности автоматически используется michael. Это основная группа пользователя michael (в некоторых дистрибутивах пользователю не выделяется отдельная группа, а все пользователи относятся к группе users).

```
michael$ touch myFile1
michael$ ls -l myFile1
-rw-r--r--    1 michael   michael      0 Jun 1416:45 myFile1
```

Пользователь michael также принадлежит к ряду других групп (команда `groups`). Чтобы создать файл, не относящийся к основной группе, сначала нужно сменить активную группу (команда `newgrp`):

```
michael$ groups
michael adm admin cdrom dokuteam dialout lpadmin plugdev sambashare
michael$ newgrp dokuteam
michael$ touch myFile2
michael$ ls -l myFile2
-rw-r--r--   1 michael    dokuteam      0 Jun 14 17:02 myFile2
```

Разумеется, `myFile2` можно было бы создать и без применения `newgrp`. В таком случае групповую отнесенность файла следовало бы постепенно изменить с помощью команды `chgrp`. Команду `newgrp` удобно использовать тогда, когда создается несколько новых файлов, которые должны автоматически присваиваться определенным группам.

Владелец и принадлежность к группе

Из двух показанных выше примеров понятно, что создаваемые файлы автоматически присваиваются своему автору. Как правило, эти файлы определяются в ту

же группу, к которой относится создавший их пользователь. Из этого правила есть два исключения.

- Если пользователь с помощью команды `newgrp` сделает активной другую группу, к которой он также принадлежит, то новый файл будет относиться к этой группе.
- Если в каталоге будет поставлен бит `Setgid` (см. предыдущий подраздел), то файлы, создаваемые в данном каталоге, автоматически попадают в ту же группу, что и этот каталог. Активная группа пользователя в расчет не принимается.

Биты доступа

С битами доступа дело обстоит немного сложнее. В Linux предусмотрено, что новые файлы получают биты доступа `rw-rw-rw` (восьмеричное значение `666`), то есть такие файлы может читать и изменять любой пользователь. Новые программные файлы, создаваемые компилятором, автоматически получают биты доступа `rwxrwxrwx` (`777`), то есть могут выполняться кем угодно.

Для практической работы с несколькими пользователями эта базовая настройка в любом случае была бы слишком бесхитростной, поэтому все оболочки Linux (то есть интерпретаторы команд) содержат так называемую umask-настройку. Она представляет собой числовое значение, указывающее биты, которые могут быть вычтены из стандартных битов доступа.

В Linux обычно применяется значение `umask`, равное `022` (`----w--w-`). Таким образом, новые файлы получают биты доступа `666-022=644` (`rw-r--r--`), новые программы — биты доступа `777-022=755` (`rwxr-xr-x`). Текущее значение настройки `umask` можно узнать (а также изменить) с помощью одноименной команды:

```
michael$ umask
0022
```

Базовая настройка значения `umask` производится в файлах конфигурации соответствующей оболочки. Что касается bash (самой популярной оболочки Linux), `umask` обычно настраивается в файле `/etc/profile` или `/etc/bashrc`. В большинстве дистрибутивов отдельные пользователи могут поставить в файле `~/.bashrc` специальную настройку, отличающуюся от задаваемой по умолчанию. Если, например, вы хотите, чтобы создаваемые вами файлы могли читать только члены группы, но не все остальные пользователи, применяйте следующую настройку:

```
# в ~/.bashrc
umask 027
```

Тогда новые файлы получают права доступа `rw-r-----`, а новые программы — `rwxr-x---`.

После того как файл будет создан, при работе другого пользователя с этим файлом не будут изменены ни данные о владельце, ни биты доступа. Изменять владельца файла может только администратор (таким образом, исключена возможность того, что владелец файла как бы «подарит» этот файл другому владельцу).

7.7. Списки контроля доступа и расширенные атрибуты

Списки контроля доступа (Access Control Lists, ACL). Типичное для UNIX управление пользователями и группами, а также основанные на нем права доступа к каталогам и файлам сохраняются в неизменном виде уже не одно десятилетие. Концепция настолько проста, что ее начинаешь понимать после пары часов изучения. Правда, встречаются случаи, когда этой простой системы недостаточно.

Именно поэтому была разработана более «мелкоячеистая» система управления правами доступа, основанная на так называемых *списках контроля доступа* (ACL). ACL позволяют установить для файла или каталога любое количество правил, касающихся того, какие пользователи или группы имеют право читать или изменять файл или группу файлов, находящихся в каталоге, а какие пользователи *не имеют такого права* — эта возможность в системах UNIX уже не предусмотрена. Таким образом, ACL дополняют стандартные права доступа и могут вводить дополнительные права или отменять существующие.

Списки контроля доступа по умолчанию входят в состав ядра Linux, начиная с версии 2.6. Для более ранних версий ядра существуют соответствующие заплатки. В файловых системах btrfs, jfs и xfs списки контроля доступа активны в любом случае. В файловых системах ext, напротив, необходимо использовать mount-параметр `acl`, чтобы активировать списки контроля доступа. В большинстве дистрибутивов эта функция не выполняется автоматически. Файловый сервер Samba — это важнейшая из тех программ, при работе с которой очень выгодно использовать списки контроля доступа. Благодаря ACL эта программа способна во многом экстраполировать права доступа, существующие в Windows, на Linux.

Ограничения. Тот факт, что ACL предоставляют дополнительные возможности, ни в коем случае не отменяет традиционного управления правами! Для опытных администраторов крупных сетей списки контроля доступа могут обеспечить дополнительную безопасность или как минимум упростить управление сетью, но для большинства пользователей Linux обычного метода управления правами будет вполне достаточно. Если неправильно использовать сложную систему списков контроля доступа, то могут возникнуть всевозможные бреши в сфере безопасности сети. Поэтому в настоящее время и не существует дистрибутивов, в которых списки контроля доступа использовались бы по умолчанию.

Основная проблема заключается в том, что многие программы и команды Linux неправильно работают с ACL. Вполне может случиться так, что в скопированном файле вдруг не окажется информации об ACL, которая присутствует в оригинале. Большинство файловых менеджеров также не могут правильно отображать и изменять списки контроля доступа (Konqueror — приятное исключение из этого правила).

Расширенные атрибуты (EA). Расширенные атрибуты (Extended Attributes, EA) находятся в тесном родстве со списками контроля доступа. Они позволяют сохранять для любых файлов дополнительные сведения в форме пар «атрибут — значение». Так, например, вы можете присвоить текстовому файлу атрибут `charset` с настройкой `utf8`, чтобы сохранить таким образом применяемую кодировку. Конечно, это выгодно только тогда, когда существуют программы, способные интер-

претировать такую информацию. В зависимости от того, какой файловой системой вы пользуетесь, расширенные атрибуты необходимо активировать с помощью соответствующего mount-параметра; например в случае с файловой системой ext это параметр user_xattr.

Более подробная информация о применении списков контроля доступа и расширенных атрибутов содержится на страницах справки man (acl, getfacl, setfacl, attr(5), getfattr и getsattr), а также на сайте http://www.vanemery.com/Linux/ACL/linux-acl.html.

Предпосылки. В следующих примерах я буду исходить из того, что у вас установлен пакет attr с командами attr, getfattr и setfattr и что вы работаете с файловой системой, в которой активированы списки контроля доступа и расширенные атрибуты. Если вы работаете с файловой системой ext4, то результат выполнения команды mount должен выглядеть так:

```
user$ mount
...
/dev/sdc5 on /test type ext4 (rw,acl,user_xattr)
...
```

В иных случаях в следующих примерах вы получите сообщение об ошибке, которое будет гласить примерно следующее: *«Операция не может быть выполнена»*. Чтобы справиться с этим, попробуйте изменить параметр mount в /etc/fstab и заново привязать файловую систему. Почитайте в главе 14 о том, как администрировать файловую систему. Информация об изменении параметров mount в /etc/fstab находится в подразделе «Автоматическое подключение файловых систем (/etc/fstab)» раздела 14.5.

Списки контроля доступа

Команда getfacl. В системах, где установлены списки контроля доступа, обычно все равно действуют стандартные принципы управления доступом, которые часто называются минимальным списком контроля доступа. Команда getfacl отображает эти права в виде ACL:

```
user$ touch файл1
user$ getfacl файл1
# file: файл1
# owner: kofler
# group: kofler
user::rw-
group::r--
other::r--
user$ ls -l файл1
-rw-r--r-- 1 kofler kofler ... файл2
```

Команда setfacl. С помощью setfacl можно определить новые права доступа. Следующие команды предоставляют пользователю gabi и всем членам группы docuteam доступ к файлу с правом читать и изменять этот файл, однако закрывают пользователю kathrin какой-либо доступ к файлу:

```
user$ setfacl -m gabi:rw файл1
user$ setfacl -m g:docuteam:rw файл1
user$ setfacl -m kathrin:- файл1
```

Список прав команды `getfacl` немного длиннее. Теперь `ls` показывает на месте прав доступа для членов группы специальную ACL-маску. За буквами, означающими доступ, следует символ +, указывающий на наличие правил ACL.

```
user$ getfacl файл1
# file: файл1
# owner: kofler
# group: kofler
user::rw-
user:gabi:rw-
user:kathrin:---
group::r--
group:docuteam:rw-
mask::rw-
other::r--
user$ ls -l файл1
-rw-rw-r--+ 1 kofler kofler ... файл1
```

Как правило, списки контроля доступа применяются для того, чтобы предоставить определенному пользователю доступ к его файлам, не давая такого доступа всем остальным пользователям. В таком случае вам потребуется попросить администратора, чтобы он создал группу, к которой будете принадлежать вы и те пользователи, вместе с которыми вы хотите обрабатывать файлы. При использовании ACL просто нужно выполнить команду `setfacl -m пользователь:rw файл`.

Маска ACL. Маска ограничивает права, предоставляемые правилами ACL. Если вы, например, устанавливаете для маски ACL значение `r`, то ни одно правило не может дать пользователю права изменять или выполнять файлы. Итак, ACL-маска имеет приоритет над правилами ACL. Но в любом случае она никак не влияет на обычные права доступа, которые предоставляются пользователю или группе пользователей традиционным способом.

При изменении правила ACL командой `setfacl` маска автоматически рассчитывается заново так, чтобы могли выполняться все прочие правила ACL. Данная маска отображается с помощью команды `getfacl` и также учитывается командой `ls -l`.

Можно явным образом настроить маску с помощью команды `setfacl -m m:rwx file`, ограничив, таким образом, права ACL. Однако учитывайте, что ваша маска действует лишь до тех пор, пока вы не определите новое правило ACL. В таком случае маска ACL будет автоматически пересчитана (если только вы не предотвратите этого с помощью параметра `-n`).

Стандартные списки контроля доступа. При работе с каталогами можно установить второй набор правил для стандартных списков контроля доступа. Стандартные ACL не управляют доступом к каталогу, а служат образцом для новых файлов. Любой файл, создаваемый внутри данного каталога, наследует стандартные списки контроля доступа, указанные для этого каталога. При одновременном применении нескольких списков контроля доступа новый каталог с правильно подо-

бранными стандартными ACL может применяться в работе в качестве отправной точки.

Совместимость ACL. Распространение списков контроля доступа сильно осложняется тем, что многие стандартные команды и практически все пользовательские программы просто игнорируют ACL. Если вы просто копируете файл с правилами ACL с помощью команды `cp`, то в скопированном файле правила ACL не сохранятся. То же самое произойдет, если вы откроете файл в текстовом редакторе OpenOffice или Gimp и сохраните его под другим именем. При использовании команды `cp` будет полезен параметр `-p`, но у большинства других команд и программ подобные параметры отсутствуют, либо эти программы вообще не приспособлены к работе с ACL.

Проблемы возникают и при резервном копировании. Команды `tar` и `rsync` элиминируют правила ACL. Файловая система CD и DVD не рассчитана на работу с ACL, так что сохраняемая в них информация теряется. Есть два выхода: во-первых, можно применять вместо `tar` версию `star`, совместимую с ACL, во-вторых, перед резервным копированием можно создавать отдельный текстовый файл, в который заносятся ACL-правила всех копируемых файлов. После резервного копирования правил ACL восстанавливаются на основании этого файла.

```
user$ getfacl -R --skip-base . > acl-backup (Сохранение ACL-правил)
user$ setfacl --restore=acl-backup          (Восстановление ACL-правил)
```

Расширенные атрибуты

Команды setfattr и getfattr. На следующих примерах показано, как сохранять атрибуты с помощью команды `setfattr` и считывать их с использованием `getfattr`.

```
user$ touch файл2
user$ setfattr -n user.language -v ru файл2
user$ setfattr --name=user.charset --value=utf8 файл2
user$ getfattr -d файл2
# file: файл2
user.charset="utf8"
user.language="ru"
```

Команда `getfattr` обычно возвращает только те атрибуты, чье название начинается с `user`. Если вы хотите увидеть другие атрибуты, нужно указывать имя атрибута вместе с меткой `-n`, а образец атрибута — с меткой `-m`.

```
user$ getfattr -n security.selinux -d tst
# file: tst
security.selinux="user_u:object_r:user_home_t:s0^000"
```

Совместимость с расширенными атрибутами. К сожалению, пока практически не существует программ, которые сохраняли бы расширенные атрибуты при копировании, архивировании и т. д. Даже команда `cp -p` игнорирует атрибуты. При создании резервных копий лучше всего поступать так же, как и при работе с ACL — создавать при копировании текстовый файл со всеми расширенными атрибутами. На основании этого файла можно потом восстановить расширенные атрибуты.

```
user$ getfattr -R . > ea-backup       (Сохранение атрибутов)
user$ setfattr --restore=ea-backup    (Восстановление атрибутов)
```

Возможности

К наиболее интересным вариантам применения расширенных атрибутов относится возможность указывать для исполняемых файлов, какие операции может совершать конкретная программа (то есть какими возможностями) она обладает. Таким образом, можно было бы помечать не так много программ Setuid-битами. И, следовательно, можно было бы значительно повысить безопасность дистрибутивов. К сожалению, ни в одном из распространенных дистрибутивов те возможности, которые будут описаны ниже, не используются сколь-нибудь активно.

Предпосылки. Чтобы можно было реализовать рассмотренные возможности, должны выполняться три предварительных условия.

- В системе должна работать достаточно актуальная версия ядра (не ниже 2.6.24), и при его компиляции должны быть активированы параметры, связанные с возможностями (CONFIG_SECURITY_CAPABILITIES и CONFIG_SECURITY_FILE_CAPABILITIES).
- Должна быть установлена библиотека libcap (/lib/libcap* или /lib64/libcap*).
- Файловая система должна поддерживать расширенные атрибуты, так как именно в виде таких атрибутов сохраняется информация о возможностях. В файловых системах на основе ext дополнительно к этому должна применяться mount-параметр user_xattr.

В большинстве дистрибутивов первые два требования по умолчанию выполняются. Для выполнения третьего условия обычно приходится внести изменения в /etc/fstab. Подробнее познакомиться с феноменом возможностей позволяют следующие сайты: http://lwn.net/Articles/313047/ и http://www.friedhoff.org/posixfilecaps.html.

Getcap и setcap. Для администрирования возможностей применяются команды getcap и setcap. Во многих дистрибутивах их приходится устанавливать дополнительно (пакет libcap-ng-utils).

В следующем примере демонстрируется применение возможностей. Сетевая команда ping в большинстве дистрибутивов снабжена Setuid-битом, поэтому она доступна для обычных пользователей. Если удалить этот бит, то с командой ping сможет работать лишь администратор (root):

```
root# chmod u-s /bin/ping
user$ ping yahoo.de
ping: icmp open socket: операция запрещена
```

Теперь можно обойтись без небезопасного бита Setuid. Достаточно с помощью setcap предоставить команде ping доступ к сетевым функциям ядра. getcap позволяет просмотреть, какие возможности теперь будут у команды ping.

```
root# setcap cap_net_raw=ep /bin/ping
root# getcap /bin/ping
/bin/ping = cap_net_raw+ep.
```

7.8. Структура каталогов в Linux

Стандарт иерархии файловой системы. Типичная UNIX-система состоит из тысяч файлов. В ходе разработки UNIX закрепились определенные правила, описывающие, в каких каталогах какие файлы принято сохранять. Эти правила были приспособлены к особенностям Linux и обобщены в специальном документе, именуемом «Стандартом иерархии файловой системы» (Filesystem Hierarchy Standard, FHS). Практически все дистрибутивы Linux за исключением нескольких построены согласно этому стандарту. Информацию о нем вы можете найти по адресу http://www.pathname.com/fhs/.

Информация, представленная в данном разделе, является лишь поверхностной справкой, помогающей сориентироваться на первом этапе (не больше!). Я учел здесь не только стандарт FHS, но и другие особенности, издавна характерные для различных популярных дистрибутивов Linux.

Файловая система начинается с корневого каталога. Как правило, в нем нет никаких файлов, только следующие каталоги.

- `/bin` — содержит элементарные команды Linux, предназначенные для управления системой. Эти команды могут выполняться любыми пользователями. Остальные программы находятся в `/usr/bin`. В современных дистрибутивах `/bin` — это просто ссылка на `/usr/bin`, то есть разница между `/bin` и `/usr/bin` практически отсутствует.
- `/boot` — хранит файлы, предназначенные для загрузки системы (обычно с помощью GRUB). В большинстве дистрибутивов здесь же располагается ядро.
- `/dev` — содержит все файлы устройств. Доступ практически к любым компонентам аппаратного обеспечения, будь то серийный интерфейс или сегмент жесткого диска, осуществляется через специальные файлы-устройства (собственно, это не настоящие файлы). Файлы-устройства динамически создаются системой `udev` (см. также ниже). В большинстве дистрибутивов каталог расположен в оперативной памяти. Поэтому содержимое этого каталога при перезапуске компьютера не сохраняется.
- `/etc` — включает в себя конфигурационные файлы для всей системы. В `/etc` имеется множество подкаталогов, в которых конфигурационные файлы распределяются по группам, например `/etc/X11` для всех X-специфичных файлов. Многие файлы из `/etc` рассмотрены в этой книге в главах, посвященным вопросам конфигурации.
- `/home` — содержит домашние каталоги всех обычных пользователей Linux. Как вы помните, домашним называется тот каталог, в котором пользователь автоматически оказывается после входа в систему и в отношении файлов которого этот пользователь имеет неограниченные права (как обычно, администратор требует отдельного упоминания — его домашний каталог называется `/root`).
- `/lib[64]` — содержит множество общих библиотек (shared libraries) или символьные ссылки на них. Эти файлы обеспечивают работу программ. Каталог `/lib/modules` включает модули ядра, которые динамически активируются или деактивируются без остановки работы системы. Остальные библиотеки находятся

в каталоге /usr/lib[64]. В каталоге /lib/firmware находятся встроенные программы (так называемая прошивка) различных компонентов аппаратного обеспечения (например, контроллер WLAN). В современных дистрибутивах /lib является ссылкой на /usr/lib. Таким образом, все библиотеки централизованно сохраняются в каталоге /usr.

- /lost+found — обычно этот каталог пуст. Если в нем есть файлы, то они являются фрагментами, которые не удалось упорядочить после восстановления файловой системы (fsck). Иными словами: были найдены сектора, но неизвестно, к какому файлу относится какой сектор. Вместо того чтобы просто удалять такие фрагменты, команда fsck копирует их в каталог /lost+found. Эта команда автоматически выполняется при запуске системы всякий раз, когда работа Linux была завершена с ошибками (перебои с электричеством, фатальный сбой и т. д.) или файловая система не проверялась в течение длительного времени. Цель fsck заключается в том, чтобы привести файловую систему в состояние, которое можно непротиворечиво описать.
- /media — содержит такие подкаталоги, как cdrom или floppy, на месте которых можно привязать внешнюю файловую систему. Раньше этот каталог назывался /mnt, но со временем закрепился вариант /media. В новых дистрибутивах появилось и новое подобное место: внешние носители данных интегрируются в каталог файловой системы /run/media/username/storagename.
- /opt — этот каталог предусмотрен для дополнительных пакетов, но в наиболее распространенных дистрибутивах он используется редко, возможно, потому, что не до конца ясно, чем дополнительные пакеты отличаются от обычных.
- /proc — хранит подкаталоги для всех процессов, выполняемых в настоящий момент. В данном случае речь не идет о настоящих файлах! Каталог /proc просто отражает внутреннее управление процессами в системе Linux.
- /root — содержит файлы root-пользователя (то есть системного администратора).
- /sbin — включает команды, предназначенные для управления системой. Общий признак всех программ, содержащихся в этом каталоге, заключается в том, что все они могут выполняться только администратором. В современных дистрибутивах /sbin является ссылкой на /usr/sbin; все команды для управления системой теперь находятся в /usr/sbin.
- /share — иногда содержит архитектурно-зависимые файлы (то есть файлы, не зависящие от процессора). Более правильно располагать такие файлы в каталоге /usr/share.
- /srv — в некоторых дистрибутивах хранит данные по серверным процессам, например /srv/www для всех документов Apache, /srv/ftp для FTP-файлов и т. д.
- /run — во многих современных дистрибутивах содержит файлы с идентификаторами процессов, а также дополнительную информацию по некоторым системным службам. Ранее такие файлы сохранялись в каталоге/var/run.

  Подкаталог /run/lock/ содержит блокирующие файлы. В более ранних дистрибутивах такие файлы находились в каталоге /var/lock.

Во многих дистрибутивах в диск оперативной памяти был перемещен либо весь каталог /run, либо как минимум некоторые его подкаталоги. В /run содержатся в основном очень маленькие файлы, которые, таким образом, больше вообще не сохраняются на жестком диске или на твердотельном накопителе. Поэтому при перезапуске компьютера они теряются.

- /sys — в версиях ядра от 2.6 и выше данный каталог содержит файловую систему sysfs. Эта система (как и proc) сообщает информацию о состоянии компьютера.
- /tmp — содержит временные файлы. Однако часто временные файлы сохраняются и в /var/tmp.
- /usr — включает все пользовательские программы, полноценную X-систему, исходные коды Linux и т. д. Как правило, содержимое этого каталога изменяется только при установке пакетов и выполнении обновлений. Изменяющиеся файлы располагаются в каталоге /var.
- /var — содержит изменяющиеся файлы. Важнейшими подкаталогами здесь являются adm (административные файлы, отличаются в зависимости от дистрибутива), lock (блокирующие файлы, предотвращающие доступ к устройствам для пользователей, не имеющих на это права), log (файлы регистрации), mail (файлы электронных сообщений, также находятся в каталоге spool/mail) и spool (сохраненные в буфере файлы для вывода на печать, новостные файлы и т. д.).

Итак, понять структуру каталогов, расположенных на корневом уровне, совсем просто. Проблемы начинаются тогда, когда каталоги /usr и /var дробятся на бесчисленные подкаталоги. В принципе многие каталоги на этом уровне называются так же, как и на корневом — лишь исполняемые программы в данном случае находятся в каталоге bin.

При этом возникает такая проблема: существует большое количество групп исполняемых программ. Это, например, текстовые команды, X-программы и т. д. Соответственно, имеется много возможностей скрывать эти приложения. Так сложилось, что с помощью ссылок в системе часто используется множество параллельных путей. Например, /usr/bin/X11 ведет к тем же программам, что и /usr/X11R6/bin (оба этих пути логически и исторически обоснованы).

Полностью описать структуру каталогов в принципе невозможно. В табл. 7.7 кратко рассматриваются подкаталоги, относящиеся к /usr.

Таблица 7.7. Каталоги из /usr

Каталог	Содержимое
/usr/bin	Исполняемые программы
/usr/games	Игры или ссылки на /usr/share/games
/usr/include	Файлы C-Include
/usr/lib[64]	Различные библиотеки, а также многочисленные подкаталоги C-компилятора, многие другие языки программирования, большие программные пакеты, например emacs и LATEX
/usr/local	Приложения и файлы, не относящиеся напрямую к дистрибутиву Linux либо установленные позже, чем основная часть системы

Продолжение ⇩

Таблица 7.7 (продолжение)

Каталог	Содержимое
/usr/sbin	Программы, исполняемые только администратором
/usr/share	Архитектурно-зависимые файлы (например, файлы Emacs-Lisp, кодировки Ghostscript и т. д.), документация (/usr/share/doc)
/usr/src	Исходный код для Linux и, возможно, для других программ

7.9. Файлы-устройства

В файловой системе Linux осуществляется управление не только файлами и каталогами, но и устройствами. Так называются особым образом обозначаемые файлы, в которых не хранится никакой информации, так как данные файлы обеспечивают связь с ядром Linux.

Старший и младший номера устройства

Устройства обеспечивают доступ ко многим компонентам аппаратного обеспечения компьютера, то есть к жестким дискам, дисководам, серийным и параллельным интерфейсам, оперативной памяти (RAM) и т. д. Три параметра — *старший номер устройства*, *младший номер устройства* и *тип доступа* (блочный или символьный) — определяют характеристику устройства.

Старший номер задает, какой драйвер ядра Linux отвечает за управление устройством. Большинство драйверов с их старшими номерами устройств описаны на сайте http://www.kernel.org/doc/Documentation/devices.txt.

У многих драйверов младший номер служит тем признаком, с помощью которого различаются отдельные (родственные) устройства: такова, например, ситуация с драйверами различных сегментов одного и того же жесткого диска.

Тип доступа указывает, являются ли устройства буферизованными (таковы все блочные устройства, например жесткие диски) или нет (символьно-ориентированные устройства, например серийные или параллельные интерфейсы).

Когда вы просматриваете содержимое каталога `/dev` с помощью команды `ls -l`, вместо размера файла система показывает номера устройства (старший и младший). Первый символ бита доступа — `b` или `c` — означает блочную либо символьную ориентацию.

```
user$ ls -l /dev/sda?
brw-rw---- 1 root root 8, 12010-02-0210:39 /dev/sda1
brw-rw---- 1 root root 8, 22010-02-0210:39 /dev/sda2
...
```

Внутренние свойства

Внутри системы Linux в каталоге `/dev` находятся только так называемые индексные дескрипторы (I-Node). Это мельчайшие управляющие единицы системы, а не настоящие файлы. Новые файлы устройств можно создавать с помощью команды

mknod. Однако на практике это требуется редко, так как система udev делает такое автоматически (см. ниже).

Старший и младший номера устройств, начиная с версии ядра 2.6, представлены в виде 64-битных чисел (до версии 2.4 для этого применялись 32-битные числа).

Доступ ко многим устройствам (из соображений безопасности) может получить только администратор либо члены определенной группы. Чтобы предоставить другому пользователю доступ к этим устройствам, добавьте его к этой группе.

Некоторые файлы устройств выполняют специфическую функцию: так, /dev/null служит «черной дырой» и отправляемые туда данные навсегда исчезают (используется, например, для переадресации командного вывода, если эта информация должна быть скрыта); /dev/zero — это неиссякаемый источник нулевых байтов, которые иногда используются для того, чтобы заполнять нулями файлы до определенного (заданного) размера; /dev/random и /dev/urandom —возвращают случайные числа (табл. 7.8).

Таблица 7.8. Важные файлы-устройства

Устройство	Значение
/dev/cdrom	Ссылка на привод CD-ROM
/dev/console	Виртуальный терминал, активный в настоящий момент
/dev/disk/*	Дополнительные ссылки на устройства жестких дисков и сегментов
/dev/dri/*	Инфраструктура прямой визуализации (3D-графика с X)
/dev/dsp*	Доступ к звуковой карте (устройству цифрового сэмплирования)
/dev/fb*	Кадровый буфер (графическая карта)
/dev/hd*	IDE-дисководы (жесткие диски, CD- и DVD-приводы)
/dev/input/*	Мышь и джойстик
/dev/kbd	Клавиатура (PS/2)
/dev/kmem	Оперативная память (RAM) с магнитным сердечником (для отладчика)
/dev/lp*	Параллельные интерфейсы для принтера и т. д.
/dev/mapper	Файлы соответствия для программы управления логическими томами (LVM), пути к контейнерам и т. д.
/dev/md*	Мета-устройства (RAID и т. д.)
/dev/mem	Память (RAM)
/dev/mixer*	Доступ к звуковой карте
/dev/psaux	Мышь PS/2
/dev/port	Порты ввода-вывода
/dev/pts/*	Виртуальные терминалы стандарта UNIX 98
/dev/ptyp*	Виртуальные терминалы для X (типа «ведущий»)
/dev/ram	Виртуальный диск
/dev/raw1394	Непосредственный доступ к Firewire-устройствам
/dev/sd*	Жесткие диски SCSI/SATA/USB/Firewire
/dev/scd*	CD/DVD-приводы типов SCSI/SATA/USB/Firewire
/dev/shm	Совместно используемая память POSIX
/dev/snd	Звук ALSA (ссылка на /proc/asound/dev)
/dev/scd*	CD/DVD-приводы типов SCSI/SATA/USB/Firewire

Продолжение ⇩

Таблица 7.8 (продолжение)

Устройство	Значение
/dev/tty*	Виртуальные терминалы для работы в текстовом режиме
/dev/ttyp*	Виртуальные терминалы для X (типа «ведомый»)
/dev/ttyS*	Серийные интерфейсы (модем, мышь и т. д.)
/dev/usb/*	USB-устройства

Система udev

Раньше дистрибутивы Linux в ходе установки создавали множество файлов устройств (например, при инсталляции Red Hat 9 создается почти 8000 таких файлов!). На деле же используется всего около сотни файлов, но этот набор неодинаков в зависимости от того, на каком компьютере установлена система и каково аппаратное обеспечение этого компьютера.

В данном случае правильный выбор помогает сделать система udev, которая появилась в версии ядра 2.6. Фоновая программа `udevd` распознает все аппаратные компоненты, подключенные к компьютеру, и создает необходимые файлы устройств. Программа `udevd` запускается в начале процесса Init-V. Конфигурация осуществляется с помощью файлов, содержащихся в каталоге `/etc/udev`.

Система udev работает просто отлично и способна обращаться с внешними винчестерами, флешками и многими другими аппаратными компонентами, которые могут подключаться к компьютеру и вновь отсоединяться в ходе работы. Основная проблема системы udev заключается в том, что создание файлов устройств при запуске компьютера длится достаточно долго (несколько секунд). Поскольку часть этих файлов необходима для продолжения запуска (в частности, для доступа к жестким дискам и сетевым интерфейсам), отложить выполнение udev или запустить ее в фоновом режиме очень сложно.

Размышления о том, как бы запускать Linux быстрее, навели разработчиков на мысль заменить udev более эффективной системой или (по крайней мере частично) вернуться к статической конфигурации. По адресу http://lwn.net/Articles/331818/ расположена статья, в которой описывается система devtmpfs, доступная, начиная с ядра 2.6.32. Остается подождать, чтобы увидеть, насколько широко она будет использоваться в дистрибутивах.

Подробные комментарии относительно называния и нумерации hd- и sd-устройств для устройств IDE- или SATA/SCSI/USB/Firewire даны в разделе 14.2. Полное описание всех устройств, имеющихся в Linux в настоящее время, вместе с соответствующими номерами вы найдете по адресу http://www.kernel.org/doc/Documentation/devices.txt.

8 Управление процессами

В данной главе описано, как в системе Linux организована работа с процессами. Здесь будут рассмотрены следующие вопросы:

- какие существуют возможности запускать программы и (при необходимости принудительно) завершать их;
- как обычному пользователю получить привилегии администратора для выполнения программы;
- что такое «демон»;
- как можно автоматически запускать программу в заданный момент времени.

8.1. Запуск программ, управление ими и завершение процессов

Программы, команды, процессы, задачи. В этой главе речь пойдет в основном о процессах. Однако практически в любом контексте можно заменить слово «процесс» терминами «программа», «команда» или «задача». Внутри системы Linux нет принципиальных отличий между программой и командой. В обиходе текстовые программы, например `ls`, часто называются «командами». Чтобы с точностью сказать, с чем мы имеем дело: с программой или командой, необходимо знать о том, какой файл при этом выполняется. Программный файл отличается от остальных файлов только тем, что имеет бит доступа `x`.

В обоих следующих файлах `server.tex` является информационным файлом, а `sichere` — программой. Точнее говоря, здесь мы имеем дело с обычным сценарием командного процессора, выполняющим резервное копирование. Оба этих файла являются текстовыми, но исполнять можно только один из них — `sichere`, так как в нем установлены биты доступа `x`.

```
user$ ls -l s*
-rw-r--r--   1 kofler   users   180383 Apr 2410:20 server.tex
-rwxr-xr-x   1 kofler   users      222 Jun  610:58 sichere
```

Только при запуске «безжизненного» программного файла он становится «живым» процессом (синоним — «задача»), а управляет этим процессом ядро Linux. В таком контексте можно было бы переформулировать название этой главы: «Запуск программ и команд, управление процессами и завершение их».

EXE-файлы. Постоянно возникает вопрос: а где в Linux находятся EXE-файлы? Еще несколько лет назад ответ звучал: «Таких файлов нет». Исполняемые программы в Linux характеризуются наличием бита доступа x. Такой файл называется исполняемым (executable). Таким образом, применяемое в Windows расширение файлов EXE оказывается ненужным.

Теперь этот ответ уже не настолько бесспорен, так как во многих системах Linux появились отдельные EXE-файлы. Это программы, написанные на языке C#, которые выполняются с помощью библиотеки Mono. Библиотека Mono — это еще одно нововведение с открытым кодом, относящееся к платформе .NET-Framework компании Microsoft.

Запуск программ

Запуск программ с X. В X программы, как правило, запускаются через меню или при щелчке кнопкой мыши на пиктограмме. В KDE и Gnome присутствует сочетание клавиш (Alt+F2), обеспечивающее быстрый запуск программ.

Текстовая консоль, командное окно. Кроме того, можно запускать программы в командном окне (например, xterm, konsole и т. д.) или в текстовой консоли. Для этого нужно просто указать название программы и нажать клавишу Enter. Профессионалы Linux работают именно таким образом, поскольку ввести пару букв гораздо быстрее, чем отыскать программу в очень разветвленном меню.

Как правило, достаточно просто указать название программы. Затем shell-интерпретатор ищет программу во всех каталогах, перечисленных в переменной окружения PATH. В следующих строках показано, как обычно настраивается эта переменная:

```
user$ echo $PATH
   /usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin:/usr/games
```

Если вы хотите запустить программу, которая находится в каком-то другом каталоге, то необходимо указать путь к ней полностью. Это касается в том числе программ текущего каталога! В таком случае путь указывается просто как точка (то есть, например, ./mojaprogramma).

Приоритетные и фоновые программы

Если запускать программы в X через меню, то они будут работать в виде так называемых фоновых процессов, не мешая друг другу. Эти процессы также могут запускать другие программы еще до завершения инициирующей программы.

Совсем иначе обстоит дело при выполнении программы в текстовой консоли или командном окне. Программа запускается как приоритетный процесс. Прежде чем запустить следующую программу, вам придется дождаться завершения работы текущего процесса.

Однако в командном окне или текстовой консоли все же можно запускать программы в фоновом режиме. Для этого после названия программы нужно просто поставить символ &:

```
user$ emacs &
```

Если вы забудете поставить &, можно постепенно перевести выполнение программы в фоновый режим. Прервите выполнение программы, нажав сочетание клавиш Ctrl+Z, а затем возобновите программу с помощью команды bg:

```
user$ emacs
<Ctrl>+<Z>
[1]+ Stopped emacs
user$ bg
[1]+ emacs &
```

Если вместо bg использовать команду fg, то программа будет выполняться как приоритетный процесс.

При выполнении некоторых программ в фоновом режиме вам будут мешать многочисленные текстовые сообщения, выдаваемые при этом. Однако избавиться от них совсем несложно, нужно просто переадресовать их в каталог /dev/null. Например, следующая команда позволяет отформатировать USB-флешку в фоновом режиме:

```
root# mkfs.ext3 /dev/sdc > /dev/null &
```

Список всех текущих процессов

Команда ps. Список всех текущих процессов создается с помощью команды ps. Без дополнительных параметров она отображает только ваши собственные процессы — и только те, которые были запущены из текстовых консолей или командных окон. Команда ps может получать разнообразные параметры. В следующем примере я сократил список процессов ради экономии места.

```
user$ ps ax
 PID  TTY    STAT   TIME  COMMAND
   1  ?      S      0:00  init [5]
   2  ?      SN     0:00  [ksoftirqd/0]
   3  ?      S      0:00  [watchdog/0]
   4  ?      S<     0:00  [events/0]
...
3064  pts/2  S      0:39  emacs command.tex
3151  pts/2  S+     1:23  /bin/sh ./lvauto
3735  pts/4  S      0:00  su -l 3740  pts/4  S+     0:00  -bash
```

Команда top. Обычно top практичнее ps: эта команда упорядочивает процессы обратно пропорционально тому, насколько они загружают процессор, и сначала отображает процессы, протекающие в настоящий момент. Кроме того, программа сообщает, какой объем памяти при этом необходим и т. д. Список процессов обновляется раз в две секунды, пока программа не завершается нажатием клавиши Q. Следующие строки характеризуют работу системы в холостом режиме:

```
top - 20:50:38 up 11 days, 12:18, 1 user, load average: 0.10, 0.09, 0.08
Tasks: 114 total, 1 running, 113 sleeping, 0 stopped, 0 zombie
Cpu(s): 2.6%us, 0.2%sy, 0.0%ni, 96.8%id, 0.4%wa, 0.0%hi, 0.0%si, 0.0%st
Mem: 4049808k total, 1580060k used, 2469748k free, 203396k buffers
Swap: 521212k total,       0k used,  521212k free, 804400k cached
```

```
PID   USER     PR NI  VIRT  RES   SHR S %CPU %MEM  TIME+  COMMAND
32601 www-data 20 0   346m  32m  4296 S    5  0.8 0:00.15 apache2
32592 www-data 20 0   344m  31m  4324 S    3  0.8 0:00.47 apache2
851   mysql    20 0  1403m  53m  7948 S    0  1.4 6:40.10 mysqld
1     root     20 0  24336 2184  1272 S    0  0.1 0:01.07 init
...
```

В столбце PID указаны номера процессов. Зная номер процесса, можно принудительно остановить вышедшие из-под контроля программы или фоновые процессы с помощью команды kill.

Процессы могут находиться в различных состояниях. Чаще всего встречаются состояния R (running)[1] и S (sleeping[2], то есть сейчас процесс не выполняет никаких задач и ожидает ввода информации). Кроме того, выполнение программы можно временно прервать, переведя ее в состояние T (stopped[3]).

Команда top обладает способностью интерактивного приема команд. При этом процессы можно останавливать (K — kill[4]) или изменять их приоритет (R).

Htop. Значительно удобнее команды top ее альтернатива — команда htop, которую в большинстве дистрибутивов требуется устанавливать отдельно. В частности, она позволяет прокручивать список процессов по горизонтали и по вертикали.

Iotop. Если вы хотите отследить не нагрузку, оказываемую на процессор и память, а доступ к жестким дискам и другим носителям данных, то запускайте вместо top команду iotop. Параметр -o позволяет ограничить вывод лишь теми процессами, которые заняты именно операциями ввода-вывода. С помощью параметра -u можно вывести на экран лишь ваши собственные процессы. Команда iotop входит в состав одноименного пакета, который, как правило, приходится устанавливать дополнительно.

Графические альтернативы top. Разумеется, для текстовой команды top существуют и графические альтернативы, например ksysguard (KDE) или gnome-system-monitor (Gnome) (рис. 8.1).

Установление номера процесса. Когда вы знаете названия программ и хотите выяснить соответствующий программе номер процесса (PID), поможет команда pidof. Если имеется несколько процессов с одинаковым названием, то pidof возвращает весь список номеров:

```
root# pidof nscd
1777 1776 1775 1774 1765 1763 1753
```

Иногда бывает полезно установить, какие программы обращаются к тому или иному файлу или каталогу. Соответствующие номера процессов можно узнать с помощью команды fuser. Каталог считается задействованным и в том случае, когда в нем была запущена программа. Следующая команда показывает, что оболочка bash использует каталог /media/dvd:

---

1 Действующий (англ.).
2 Спящий (англ.).
3 Остановленный (англ.).
4 Убить (англ.).

```
root# fuser -v /media/dvd
                     USER        PID  ACCESS  COMMAND
/media/dvd           kofler     2183  ..c..   bash
                     Root     Kernel  mount   /media/dvd
```

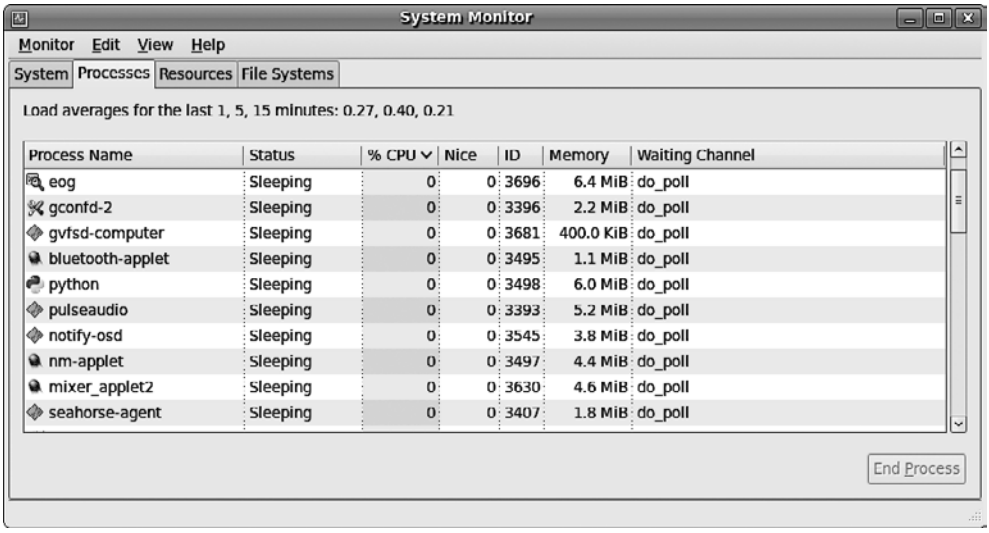


Рис. 8.1. Обзор процессов с помощью программы gnome-system-monitor

Обратите внимание, что факт доступа к файлу может быть установлен лишь в том случае, когда программа действительно открыла файл на достаточно долгий период времени. В случае с текстовым редактором этого, например, не происходит (текстовый редактор открыл файл для загрузки, но потом закрыл; чтобы сохранить файл, он вновь ненадолго его открывает).

PID-файлы. Некоторые фоновые процессы сохраняют в каталоге `/var/run` PID-файл. В первой строке этого файла содержится номер процесса, а в остальных строках может находиться дополнительная информация, например данные о сетевом интерфейсе. PID-файлы обеспечивают целенаправленное завершение определенного процесса с помощью системы Init, в том числе и тогда, когда существует несколько одноименных процессов.

Иерархия процессов

Внутри системы вместе с каждым процессом сохраняется PID-номер его родительского процесса. Эта информация позволяет построить дерево процессов, на вершине которого всегда располагается процесс `init`. Это первая программа, запускаемая сразу же после загрузки ядра.

Увидеть иерархию процессов можно с помощью команды `pstree`. Параметр `-h` позволяет выделить те процессы, которые являются родительскими для процесса, выполняемого в настоящий момент (рис. 8.2).

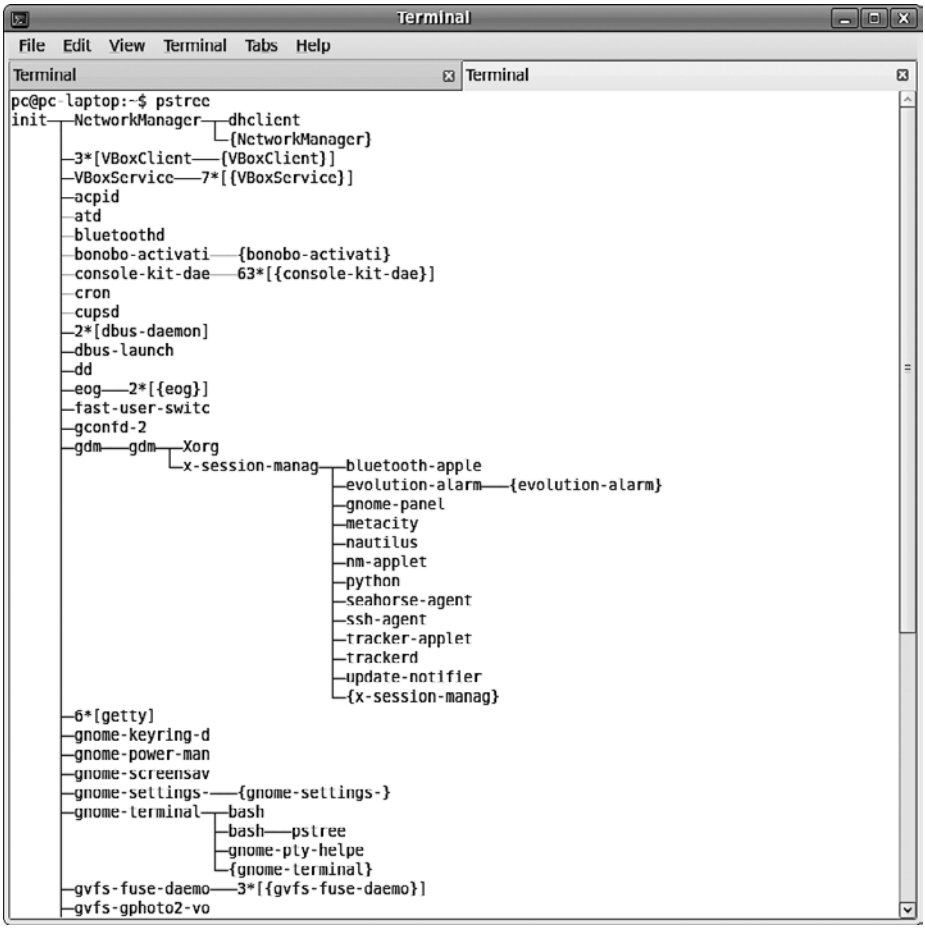


Рис. 8.2. Обзор процессов с помощью команды pstree

Принудительное завершение процессов

Как правило, с окончанием программы завершается и процесс. Но, к сожалению, и в Linux программы могут содержать ошибки, не позволяющие им остановиться, из-за чего такие процессы все сильнее потребляют ресурсы оперативной памяти и процессора. В таких случаях процесс необходимо принудительно завершать. При работе с текстовыми командами для принудительного завершения достаточно нажать Ctrl+C. Программа сразу завершается.

Команда kill. Эта команда посылает действующему процессу сигналы, специфицируемые благодаря номеру `PID` (его можно узнать с помощью команды `top` или `ps`). Чтобы «аккуратно» завершить программу, используется сигнал 15 (`kill` задействует этот сигнал по умолчанию). Если это не помогает, необходимо применить сигнал 9 (в данном случае — для процесса 2725):

```
user$ kill -92725
```

Команду kill можно применять только к собственным процессам. Администратор имеет право завершать процессы и других пользователей.

Команда top. Можно завершать процессы и с помощью команды top: просто нажмите клавишу K и дополнительно — номер процесса, а также желаемый сигнал.

Команда killall. Эта команда гораздо удобнее, так как при ее использовании можно указывать не номер процесса, а название программы. Правда, в данном случае будут завершены все процессы с таким именем.

```
root# killall -9 firefox
```

Команда xkill. В X все еще удобнее. Запустите в командном окне команду xkill и просто щелкните кнопкой мыши в окне с программой, которую хотите завершить. Процессу также будет отправлен сигнал 9.

В KDE можно запускать xkill и с помощью сочетания клавиш Ctrl+Alt+Esc. Если вы нажмете его по ошибке, то можно прервать выполнение xkill с помощью клавиши Esc.

Сложные случаи. Иногда xkill закрывает окно, а процесс или его части продолжают работать. Убедитесь в том, что команда действительно прекратила работу, выполнив команду top или ps. При необходимости воспользуйтесь командой kill -9 n.

Блокировка мыши и клавиатуры. И уж совсем неприятно, когда X-программа не просто остается висеть, но и переводит на себя фокус мыши или клавиатуры либо каким-то образом блокирует X. При этом компьютер почти не реагирует на ввод. В таких случаях иногда помогает волшебное сочетание Ctrl+Alt+F1, позволяющее перейти в первую текстовую консоль. Оттуда можно войти в систему, найти зависшую программу и завершить ее с помощью команды top.

Когда клавиатура полностью заблокирована, всегда остается возможность зайти в систему из сети через SSH и выполнить kill таким образом. Разумеется, этот вариант возможен лишь тогда, когда вы работаете в локальной сети, а на локальном компьютере запущена программа sshd.

Если сама X окажется заблокированной после остановки программы, попытайтесь принудительно завершить работу или, наконец, выполнить команду shutdown. Все эти варианты лучше, чем нажатие Reset, так как в последнем случае можно потерять данные!

Ограничение размера процесса. При работе с программами, запускаемыми из оболочки (например, со всеми командами, которые выполняются в командном окне), можно воспользоваться оболочковой командой ulimit, которая позволяет ограничить максимальный размер потребляемой памяти, создаваемых файлов и т. д. Обычно ulimit настраивается с помощью файла /etc/profile.

Распределение машинного времени (продолжительности вычислений)

При повседневной работе с Linux вычислительной мощности машины обычно более чем достаточно, чтобы выполнять все текущие процессы без задержек. Однако, если Linux занята выполнением работы, требующей большого объема

вычислений, например компилирует крупную программу, то система пытается справедливо распределить имеющееся в распоряжении вычислительное время между всеми процессами.

Иногда бывает целесообразно выделить определенному процессу заметно больше или несколько меньше времени, нежели остальным. Для этого предназначена команда `nice`, с помощью которой можно запускать программы с повышением или понижением приоритета. В таком случае команде `nice` сообщается приоритет, значение которого находится в диапазоне между `19` (совсем низко) и `-20` (очень высоко). По умолчанию процессы запускаются с приоритетом `0`. В следующем примере программа резервного копирования запускается с низким приоритетом, чтобы она не влияла на выполнение других процессов (ведь ничего страшного не случится, если резервное копирование продлится на три секунды дольше).

```
user$ nice -n 10 sichere
```

Renice. С помощью команды `renice` можно изменять приоритет процессов, выполняемых в настоящий момент. В качестве параметра необходимо указать ID процесса (который вы ранее узнали с помощью команды `top` или `ps`). Подробная информация о `renice` содержится на соответствующей странице справки `man`. Команда `top` также может интерактивно менять приоритет процесса. В любом случае только администратор может выполнять программы с более высоким приоритетом, чем `0`, либо повышать приоритет процесса, который уже выполняется.

Ionice. На компьютерах с многоядерными процессорами при выполнении программ лимитирующим фактором зачастую оказывается размер жесткого диска. Если вы хотите избежать таких ситуаций, в которых, например, сценарий резервного копирования затребует всю мощность ввода-вывода компьютера и затормозит таким образом другие процессы, критические по времени, то можно выполнять `ionice` со сниженным приоритетом ввода-вывода. Следующая команда считывает логический том, архивирует его содержимое и сохраняет его в файле образа:

```
root# ionice -c 3 cat /dev/vg1/snap | lzop -c > /backup/image.lzo
```

Многопоточность. Linux не только может параллельно выполнять несколько процессов, но и различает в пределах одного процесса отдельные подпроцессы (потоки). Способность подразделять функции на несколько потоков характерна, прежде всего, для серверных приложений. Это помогает значительно повысить производительность, особенно тогда, когда в распоряжении имеется несколько процессоров или многоядерный процессор. При управлении потоками и обменом информацией между ними помогают функции ядра. Начиная с версии ядра 2.6, библиотека Native POSIX Thread Library поддерживает потоки в соответствии со стандартом POSIX.

Переадресация ввода и вывода, программный канал

Практически все текстовые программы (команды) ожидают ввода через так называемый стандартный канал ввода (по умолчанию это клавиатура) и посылают

результаты работы через стандартный канал вывода (текст отображается в консоли или в командном окне). И ввод, и вывод можно переадресовывать, благодаря чему открываются богатые дополнительные возможности. Например, следующая команда сохраняет список всех файлов, находящихся в каталоге xy, в файл z:

```
user$ ls xy > z
```

С помощью так называемых программных каналов вывод одной команды может быть интерпретирован как данные ввода для другой команды. В следующем примере команда grep отфильтровывает из списка всех установленных пакетов те, в которых содержится последовательность символов mysql без учета регистра. Команда sort сортирует полученный список.

Иными словами, благодаря использованию программных каналов (обозначаемых символом |) вывод команды rpm переадресовывается команде grep, а вывод этой команды переадресовывается команде sort (второй символ |). Более подробное объяснение и примеры переадресации ввода и вывода, а также информация об использовании программных каналов приводится в разделе 6.4.

```
user$ rpm -qa | grep -i mysql | sort
mysql-5.1.32-1.fc11.i586
mysql-connector-java-3.1.12-7.fc11.i586
...
```

8.2. Выполнение процессов от имени другого пользователя (su)

При выполнении программы обычным пользователем действуют два ограничения.

- Рядовые пользователи могут выполнять только такие процессы, в которых это допускается в соответствии с правами доступа (пользователь, группа, биты доступа r и x). При работе с обычными программами это ограничение никак не проявляется. Но, например, в каталоге /usr/sbin есть отдельные команды, предназначенные для системного администрирования, которые могут запускаться только администратором.
- Как я уже говорил, процессы принадлежат тому пользователю, который их запустил. Это означает, что процесс имеет доступ к тем же файлам, что и пользователь, запустивший его (то есть ваши программы не имеют доступа к тем файлам, которые вы как пользователь не имеете права изменять). Новые файлы, созданные процессом, также принадлежат пользователю, запустившему программу.

По этим причинам обычный пользователь не может выполнять многие административные задачи. Очевидно, что самое простое решение — войти в систему с правами администратора. Однако, как я уже не раз указывал, не стоит постоянно работать в качестве администратора: слишком велика опасность случайного причинения вреда. Поэтому в некоторых дистрибутивах в принципе отключена возможность входа с правами администратора. Так, например, в Ubuntu, непосредственный вход в систему с правами администратора невозможен.

В этом разделе рассказывается, как можно выполнять административные задачи с помощью su и ssh, оставаясь в статусе обычного пользователя. В следующем разделе мы рассмотрим альтернативный метод с sudo, закрепившийся, главным образом, в Ubuntu. Наконец, в разделе 8.4 будет описана программа PolicyKit, предоставляющая принципиально новые возможности выполнения административных задач.

Как обычно, я мог бы рассказать гораздо больше, чем позволяют размеры этой книги. Более подробная информация содержится в документах Security-HOWTO и Remote-X-Mini-HOWTO, расположенных по адресам http://www.tldp.org/HOWTO/Security-HOWTO/index.html и http://www.tldp.org/HOWTO/Remote-X-Apps.html соответственно. Оба этих документа уже достаточно старые, но содержащиеся в них советы по-прежнему сохраняют актуальность.

Команда su. Часто требуется получить привилегии администратора в первую очередь для того, чтобы быстро выполнить команду, — покидать ради этого X было бы очень неудобно. Самая простая возможность изменить пользователя, не покидая окна X-Shell, — воспользоваться командой su name. Если вы выполняете эту команду без прав администратора, то система запросит у вас пароль соответствующего пользователя. После ввода пароля вы сможете выполнять команды в командном окне (xterm, konsole) под измененным именем, пока не вернетесь в обычный режим с помощью команды exit или сочетания клавиш Ctrl+D.

В следующих строках показано, как обычный пользователь ненадолго входит в систему с правами администратора, пользуясь этими правами, осуществляет привязку сегмента жесткого диска дереву каталогов, а затем выходит из системы (как администратор) и продолжает работать в обычном режиме:

```
user$ su -l root
Password: xxx
root# mount -t ext2 /test /dev/hdc7
root# <Ctrl>+<D>
logout
user$ ls /test
```

Чтобы команда su была полноценной заменой для входа в систему в качестве администратора, необходимо пользоваться параметром -l! Таким образом гарантируется, что стартовые файлы, обеспечивающие вход в систему (например, необходимые для определения PATH), будут прочитаны.

Команда kdesu. В KDE для запуска программ X лучше всего использовать команду kdesu. Программа показывает окно для пароля администратора.

```
user$ kdesu kate /etc/fstab
```

Команда kdesu действует лишь в том случае, когда запущен демон kdesud. Обычно этот демон запускается вместе с KDE. Обобщенная справка по всем параметрам kdesu предоставляется командой kdesu -helpall. В некоторых дистрибутивах kdesu интегрирована прямо в меню KDE.

Итак, когда вы открываете программу, требующую привилегий администратора, на экран автоматически выводится форма входа в систему, управляемая kdesu.

Команда gksu. Аналог kdesu из системы Gnome называется gksu. Эта команда работает без дополнительного фонового процесса. Обобщенная справка по параметрам команды дается в man gksu.

Сценарий su-to-root. Debian (но только не Ubuntu!) использует для запуска административного инструментария из меню Gnome или KDE независимый от локального компьютера сценарий su-to-root. Он входит в состав пакета menu. Справка man su-to-root обобщает немногочисленные параметры данного сценария.

Команда consolehelper. В Fedora и Red Hat используется разновидность gksu под названием consolehelper. Эта программа также открывает окно, но работает совершенно по-другому. Основная идея заключается в том, что применяемый инструментарий администратора устанавливается в каталог /usr/sbin, где этими инструментами может пользоваться только администратор. Для обычных пользователей в каталоге /usr/bin предусмотрена символьная ссылка на consolehelper. Следующая команда демонстрирует соответствующую конфигурацию system-config-network (сетевая конфигурация для Red Hat или Fedora):

```
user$ ls -l /usr/sbin/system-config-network /usr/bin/system-config-network
-rwxr-xr-x ... root root /usr/sbin/system-config-network
lrwxrwxrwx ... root root /usr/bin/system-config-network -> consolehelper
```

Если теперь администратор выполнит команду system-config-network, то запустится механизм /usr/sbin/system-config-network. Напротив, если обычный пользователь выполнит system-config-network, то запустится consolehelper. Если пользователь укажет правильный пароль администратора, то consolehelper запустит нужную программу system-config-network.

Программа ssh. В большинстве дистрибутивов su работает только с текстовыми командами. Для этого может быть много причин: во-первых, для запуска X-программы требуется переменная окружения DISPLAY, которая должна содержать имя компьютера, где будет отображаться программа (export DISPLAY=localhost:0). Во-вторых, из соображений безопасности X может запретить пользователям со стороны запускать программы (в данном случае вам пригодится команда xhost). В-третьих, сетевой порт может быть закрыт для обмена информацией с X-сервером.

Если у вас в распоряжении нет kdesu, gksu или consolehelper, то ssh предлагает простейшее решение всех рассмотренных проблем: просто выполните нужную команду по следующему образцу:

```
user$ ssh -X -l user localhost
```

Биты доступа Setuid и Setgid. Биты доступа Setuid и Setgid предлагают еще одну возможность помечать определенные программы так, чтобы любой пользователь мог их выполнять, как если бы он был администратором. Важнейшее отличие от sudo заключается в том, что биты доступа setuid и setgid действительны для *всех* пользователей (а при работе с sudo таких пользователей нужно явно перечислить). Более подробная информация по битам доступа suid и guid находится в пунктах о «Бит Setuid» и «Бит Setgid» подраздела «Специальные биты» раздела 7.6.

8.3. Выполнение процессов от имени другого пользователя (sudo)

Программа sudo применяет при работе принципиально иной подход по сравнению с описанными выше вариантами su. После проведения конфигурации она предоставляет определенным пользователям права администратора для выполнения определенных программ. Из соображений безопасности нужно указывать при этом *свой* пароль (а не пароль администратора!).

В таком случае sudo выполняет соответствующие программы так, как если бы они были запущены другим пользователем (по умолчанию администратором). Таким образом, отдельные пользователи могут брать на себя выполнение административных задач либо выполнять команды, критически важные для работы системы, не зная при этом пароля администратора. Программа sudo протоколирует все выполненные команды (в том числе неудачные попытки) и обычно заносит эту информацию в файл /var/log/messages.

Программа sudo запоминает пароль на 15 минут. Если в течение этого времени вы выполните с помощью sudo другую команду, вам не будет направляться запрос о пароле (заданное время можно изменить в файле /etc/sudoers с помощью ключевого слова timestamp_timeout).

Конфигурация. Конфигурация sudo осуществляется с помощью файла /etc/sudoers. Проще говоря, в этом файле в трех столбцах описывается, какие пользователи с каких компьютеров имеют право выполнять отдельные программы. Следующая запись означает, что пользователь kathrin, работающий на компьютере uranus, имеет право выполнять команду /sbin/fdisk (ключевое слово ALL означает, что kathrin имеет право выполнять команду под любой учетной записью, то есть как root, news, lp и т. д.).

```
# в /etc/sudoers
kathrin uranus=(ALL) /sbin/fdisk
```

Если в первом столбце перед sudoers поставить символ %, то запись будет действительна для всех членов указанной группы. Многочисленные прочие варианты синтаксиса описаны в файле справки man sudoers.

ВНИМАНИЕ

Из соображений безопасности рекомендую редактировать /etc/sudoers исключительно с помощью visudo (см. также соответствующий раздел справки!) Перед сохранением visudo проверяет синтаксис и гарантирует, что вы не совершили ошибок, которые могли бы не позволить вам выполнять другие административные задачи. Это особенно важно в таких дистрибутивах, как Ubuntu, где не предусмотрен вход в систему с привилегиями администратора.

Применение. Пользователь kathrin теперь может выполнить команду fdisk следующим образом. Вводим пароль пользователя kathrin. При работе с fdisk нужно указать путь целиком, если программа не находится в одном из каталогов PATH пользователя kathrin. Команда fdisk будет автоматически выполнена с привилегиями администратора. Чтобы выбрать другую учетную запись для выполнения, нужно указать ее так: sudo -u *учетная запись*.

```
kathrin$ sudo /sbin/fdisk /dev/sda
Password: xxxxxx
```

Sudo без пароля. Можно позволить определенному пользователю выполнять sudo без указания пароля. Для этого внесите в sudoers строку, построенную по следующему образцу:

```
kofler ALL=(ALL) NOPASSWD: ALL
```

Конечно, такой метод небезопасен, но если вам приходится часто выполнять административные задачи, то вы по достоинству оцените этот удобный способ. Обратите внимание, что метка NOPASSWD действительна лишь в том случае, когда отсутствуют другие строки sudoers, требующие, чтобы пользователь указал пароль. Это же правило действительно для групповых записей, то есть, например, %admin ....

Выполнение нескольких команд с помощью sudo. При решении объемных административных задач со временем станет обременительно ставить перед каждой командой слово sudo. Гораздо более элегантное решение — перейти с помощью sudo -s в режим администратора. Все последующие команды будут выполняться с правами администратора. Чтобы выйти из этого режима, нажмите сочетание клавиш Ctrl+D.

Программа gksudo. При запуске административных программ в X во многих дистрибутивах используется gksudo. Эта программа позволяет ввести пароль в диалоговом окне, а затем запускает желаемую программу.

Link. Конфигурация /etc/sudoers предусматривает гораздо больше синтаксических вариантов, чем описано выше. Обязательно прочтите страницы man-справки по sudo и sudoers! Еще подробнее о sudo рассказано на сайте программы: http://www.courtesan.com/sudo/.

Sudo в Ubuntu

В Ubuntu и некоторых других дистрибутивах пароль администратора не предусмотрен. Поэтому войти в систему с привилегиями администратора невозможно. Команды su или ssh -l root также не работают (правда, функционирует sudo su -l). В данном случае можно выполнять административные команды только с помощью sudo. В файле /etc/sudoers есть всего несколько строк:

```
# Конфигурация по умолчанию в /etc/sudoers в Ubuntu
Defaults  env_reset
Defaults  mail_badpass
Defaults  secure_path=
          "/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin"
root      ALL=(ALL) ALL
%admin    ALL=(ALL) ALL
%sudo     ALL=(ALL:ALL) ALL
```

Строка Defaults env_reset возвращает все переменные окружения в исходные значения в случае смены пользователя. Строка Defaults mail_badpass приводит к тому, что после ошибочной попытки входа в систему администратору отправляется

предупреждение. Строка `Defaults secure_path` задает содержимое переменной окружения `PATH` для команды `sudo`.

Четвертая строка предоставляет администратору неограниченный доступ к любым программам. В Ubuntu эта строка в принципе является бессмысленной, так как вход в систему с правами администратора невозможен. Важнее всего последние две строки: они позволяют всем членам групп sudo и admin обращаться к любым программам.

В Ubuntu по умолчанию членом группы sudo является только первый пользователь (тот, кто проводил установку). Если вы создаете в системе учетные записи других пользователей, то их также можно определить в эту административную группу. В сравнительно старых версиях Ubuntu вместо группы sudo использовалась группа admin.

Следующая дополнительная строка в `/etc/sudoers` позволяет пользователю kofler выполнять без пароля команду `apt-get` и программу Synaptic. Таким же образом можно устанавливать обновления, не указывая пароль.

```
# Дополнение в /etc/sudoers в Ubuntu
kofler ALL=NOPASSWD: /usr/sbin/synaptic, /usr/bin/apt-get
```

Sudo в SUSE

В SUSE роль `sudo` значительно скромнее, чем в Ubuntu. Если вы все же хотите воспользоваться `sudo`, обратите внимание на некоторые особенности конфигурации, задаваемой по умолчанию (при необходимости такие установки можно изменить).

```
# Конфигурация, задаваемая по умолчанию в /etc/sudoers
# в дистрибутиве openSUSE
Defaults always_set_home  # Запрещает выполнять X-программы
Defaults env_reset        # Запрещает выполнять X-программы
Defaults env_keep = "LANG LC_ADDRESS LC_CTYPE ..."
Defaults targetpw         # sudo запрашивает пароль целевого пользователя
ALL ALL=(ALL) ALL         # Если пароль правильный, то всем можно все
root  ALL=(ALL) ALL
```

Первые три строки из соображений безопасности не позволяют выполнять X-программы с помощью `sudo`. Если `sudo` должна поддерживать непосредственный запуск таких программ, эту строку нужно удалить или поставить перед ней символ #.

Выражение `Defaults targetpw` означает, что в принципе необходимо указать пароль для учетной записи, с которой будет выполняться команда (как правило, это пароль администратора). Наконец, строка `ALL ALL=(ALL) ALL` позволяет любым пользователям выполнять любые команды, если известен пароль нужной учетной записи.

8.4. Выполнение процессов от имени другого пользователя (PolicyKit)

Концепция. Основная концепция PolicyKit заключается в том, что при работе программы подразделяются на два компонента: первый содержит пользовательский

интерфейс и работает с обычными пользовательскими правами. Вторая часть программы, именуемая в номенклатуре PolicyKit *«механизм»*, предназначена для вмешательства в функционирование системы и работает с правами администратора. Такое разделение связано с принципиальным преимуществом — требуется запускать не огромную программу, обладающую правами администратора, а всего лишь небольшие компоненты. Это снижает потенциальный риск для безопасности системы. Кроме того, теоретически существует возможность того, что различные пользовательские интерфейсы (например, программы для Gnome и KDE) смогут обращаться к одинаковому набору механизмов.

Обмен информацией между обоими компонентами осуществляется через систему шин (как правило, через шину D-Bus). Решение о возможности выполнения того или иного механизма принимается функциями, которые содержатся в библиотеке PolicyKit и обращаются к центральной базе данных, где хранится информация о правах. При принятии решения учитываются три критерия:

- **субъект** — кто (какой пользователь) хочет внести изменения в систему;
- **объект** — какой объект требуется изменить (например, файл, сегмент диска или сетевое соединение);
- **действие** — что должно быть сделано (например, нужно привязать сегмент к файловой системе).

С точки зрения пользователя. Во многих случаях пользователь даже не замечает работы PolicyKit. Например, стандартная конфигурация большинства дистрибутивов позволяет связывать с файловой системой файловый менеджер и внешние носители данных. Поскольку дальнейшая аутентификация не требуется, процесс протекает автоматически, как только вы подсоедините носитель данных.

При использовании второго варианта, например при обновлении системы с помощью PackageKit, по правилам PolicyKit требуется авторизация. Для этого выводится специальное окно аутентификации. Следует отметить, что при определенной конфигурации PolicyKit запоминает данные, введенные в ходе аутентификации, и сохраняет их в файлах `*.auths` в каталоге `/var/lib/PolicyKit/`. Иначе говоря, если пользователь однажды войдет в систему, чтобы выполнить определенные операции, PolicyKit больше не будет запрашивать авторизацию.

Третий вариант используется при работе с различными административными инструментами Gnome: здесь кнопка Разблокировать открывает окно аутентификации. Только после указания пароля пользователя или администратора можно будет внести изменения в систему.

Конфигурация. Конфигурация PolicyKit производится в трех следующих местах:

```
/etc/polkit-1/*                      (Глобальная конфигурация)
/usr/share/PolicyKit/policy/*.policy (Действия)
/var/lib/polkit-1/*                  (Права)
```

Базовая конфигурация имеет отличительные особенности, характерные для конкретных дистрибутивов. Например, в системах Ubuntu файл `/etc/polkit-1/localauthority.conf.d/51-ubuntu-admin.conf` дает всем пользователям групп admin и sudo права администратора. В Fedora аналогичные права предоставляются членам группы wheel.

8.5. Системные процессы (демоны)

Демоны (daemons) — это фоновые процессы, предназначенные для управления системой (табл. 8.1). Обычно эти процессы запускаются при включении компьютера в рамках процесса Init-V. Если вы хорошо знакомы с терминологией Windows, то подскажу: *демоны* Linux соответствуют *службам* Windows.

По мере описания программ на страницах этой книги я буду давать ссылки на посвященные им сайты. Однако не забывайте, что названия демонов могут отличаться от дистрибутива к дистрибутиву (например, `httpd` или `apache2` — демон веб-сервера Apache).

Таблица 8.1. Важные системные процессы

Процесс	Значение
afpd	Сервер для Apple Filing Protocol
apache2	Веб-сервер
atd	Запуск других программ в заданное время (подобно cron)
avahi-daemon	Автоматическая конфигурация сети (ZeroConf, Rendezvous, Bonjour)
bluetoothd	Управление технологией Bluetooth
cron	Запуск других программ в заданное время
cupsd	Диспетчер печати
dbus-daemon	Обмен данными через шину D-BUS
dhclient	DHCP-клиент
dhcpd	Присваивание другим компьютерам сетевого IP-адреса
dhcpcd	Считывание сетевого IP-адреса
gdm	Менеджер учетных записей Gnome
gpm	Управление мышью для текстовых консолей
hpiod	Функции печати и сканирования HP
httpd	Веб-сервер (например, Apache)
kdm	Менеджер учетных записей KDE
lockd	Функция NFS-Locking
lpd	Обычный диспетчер печати на основе BSD-LPD
mdnsd	Автоматическая конфигурация сети (ZeroConf, Rendezvous, Bonjour)
mysqld	Сервер базы данных MySQL
named	Сервер доменных имен
NetworkManager	Устройство управления сетью
nmbd	Сервер имен для Windows/Samba
nscd	Кэш для хранения информации о пользователях, группах и именах компьютеров
ntpd	Установка времени с помощью сетевого протокола синхронизации времени (Network Time Protocol)
pppd	VPN-клиент, доступ к Интернету
pptpd	PPTP-сервер для VPN
portmap	Часть NFS-сервера
postfix	Почтовый сервер для рассылки электронных сообщений
rpc.*	Сетевые службы RPC (удаленного вызова процедур), обычно для NFS

Процесс	Значение
sdpd	Управление технологией Bluetooth
sendmail	Почтовый сервер для рассылки электронных сообщений
smartd	Система оценки состояния жесткого диска SMART
smbd	Файловый сервер Windows/Samba
squid	Сетевые прокси и сетевой кэш
sshd	Сервер SSH
rsyslogd	Протоколирование системных сообщений
udevd	Управление устройствами
vsftpd	FTP-сервер
xdm	Дисплейный менеджер для X
xinetd	Запуск других сетевых демонов

Потоки ядра

Кроме обычных серверных служб, например `httpd` (Apache), существуют фоновые процессы, которые, однако, являются не настоящими программами, а потоками (threads) ядра. Эти процессы можно опознать по тому, что рядом с их названиями в квадратных скобках записывается `ps axu`. Некоторые из таких потоков сопровождаются номером, указывающим на процессор. Таким образом, `kblockd/0` управляет буфером блочного устройства для первого процессора, `kblockd/1` — для второго процессора и т. д.

Большинство потоков ядра занимаются выполнением низкоуровневых задач операционной системы (управление памятью, процессами, ЦПУ и т. д.). Многие потоки запускаются уже в ходе инициализации системы в начале ее старта. Для выполнения обычных требований дополнительного конфигурирования не требуется. Функции важнейших потоков ядра приведены в табл. 8.2.

Таблица 8.2. Важные потоки ядра

Поток ядра	Значение
aio	Асинхронное управление вводом/выводом (например, для сетевых процессов)
events	Управление событиями и программными прерываниями
kacpid	Функции ACPI (усовершенствованного интерфейса конфигурации и управления питанием)
kblockd	Управление буфером блочных устройств
khelperd	Загрузка или удаление модулей ядра для пользовательских программ
khubd	Управление подключением и извлечением USB-устройств
kjournald	Осуществление журналирования для файловых систем ext3/4
knfsd	NFS-сервер
kthread	Управление потоками
nfsd	NFS-сервер
kscand	Управление памятью

Продолжение ⇩

Таблица 8.2 (продолжение)

Поток ядра	Значение
kseriod	Обмен информацией с серийными устройствами
ksoftirqd	Управление аппаратными прерываниями
kswapd	Свопинг
lockd	NFS-Locking
migration	Определение, какие процессы выполняются на каком процессоре
pdflush	Физическое сохранение изменений, вносимых в файлы
rpciod	NFS
scsi_eh	Управление ошибками и перерывами SCSI
watchdog	Проверка того, реагирует ли еще система на команды

Запуск и завершение работы демонов

В большинстве дистрибутивов вышеперечисленные демоны запускаются системой Init-V. Базовая и подробная информация по этой системе приводится в главе 16. Там же будет рассказано, как вы можете интегрировать в систему новые сценарии.

Здесь я очень сжато объясню, как можно запустить или остановить демон вручную, что сделать, чтобы демон автоматически запускался при старте системы, либо как избежать автоматического запуска демона. Эта информация особенно пригодится вам при создании и конфигурировании сетевых служб. Обратите внимание, что в разных дистрибутивах различаются не только команды, но и названия служб. Например, в Debian, Ubuntu и SUSE сценарий для запуска веб-сервера Apache называется `apache2`, а в Fedora и Red Hat — `httpd`.

Запуск вручную

Чтобы запустить демон, сетевую службу или сервер, выполните одну из следующих команд. `service` работает в большинстве распространенных дистрибутивов за исключением Debian. Доступность остальных команд зависит от конкретного дистрибутива, соответственно, от системы Init-V.

```
root# service имя start            (Функционирует почти во всех дистрибутивах)
root# /etc/init.d/имя start        (Процессы Init-V)
root# invoke-rc.d имя start        (Работает с Init-V в Debian)
root# rcимя start                  (Работает с Init-V в SUSE)
root# start имя                    (Upstart, например в Ubuntu)
root# systemctl start имя.service  (Systemd, например в Fedora и openSUSE)
```

Остановка вручную

Теперь рассмотрим команды, предназначенные для остановки служб:

```
root# service имя stop             (Почти во всех современных дистрибутивах)
root# /etc/init.d/имя stop         (Процессы Init-V)
root# invoke-rc.d имя stop         (Работает с Init-V в Debian)
root# rcимя stop                   (Работает с Init-V в SUSE)
root# stop имя                     (Upstart)
root# systemctl stop имя.service   (Systemd)
```

Перезагрузка/перезапуск

Многие сетевые службы позволяют заново считывать конфигурационную информацию, не останавливая работу системы. Команда `reload` нужна для того, чтобы служба учитывала изменения, внесенные в конфигурационный файл. Службы, которые не поддерживают `reload`, следует полностью перезапускать с помощью команды `restart`.

```
root# service имя reload/restart      (Почти во всех современных дистрибутивах)
root# /etc/init.d/имя reload/restart  (Процессы Init-V)
root# invoke-rc.d имя reload/restart  (Работает с Init-V в Debian)
root# rcимя reload/restart            (Работает с Init-V в SUSE)
root# reload/restart имя              (Upstart)
root# systemctl reload имя.service    (Systemd)
```

Автоматический старт при запуске компьютера

Когда вы правильно настроите сетевую службу, она будет запускаться автоматически при загрузке компьютера, а при завершении работы системы служба будет автоматически останавливаться. В некоторых дистрибутивах (например, Debian, Ubuntu) службы конфигурируются требуемым образом уже при установке пакетов. в других дистрибутивах этого не происходит по причинам, связанным с безопасностью, и автоматический старт требуется активировать вручную.

В Debian версии 6 и выше, а также в SUSE команда `insserv` отвечает за создание ссылок для запуска и остановки Init-V.

В Red Hat и Fedora обычно бывает достаточно первой команды `chkconfig`. Вторая команда требуется лишь в тех случаях, когда сценарий Init-V не содержит никаких указаний относительно того, на каком уровне (runlevel) запускается служба (обычно это уровни 3 и 5).

```
root# insserv имя                  (Специфично для Init-V в Debian и SUSE)
root# chkconfig --add имя          (Специфично для Init-V в Red Hat)
root# chkconfig --level 35 имя on  (Специфично для Init-V в Red Hat)
root# systemctl enable имя.service (Systemd)
```

Предотвращение автоматического запуска

Следующие команды позволяют предотвратить автоматический запуск службы при старте системы. Если служба уже работает, то ее нельзя остановить какой-либо из следующих команд; для этого требуется специальная команда `stop`.

```
root# insserv -r имя                  (Специфично для Init-V в Debian и SUSE)
root# chkconfig --del имя             (Специфично для Init-V в Red Hat)
root# systemctl disable имя.service   (Systemd)
```

При работе с системными службами, запускаемыми с помощью Upstart (Ubuntu, Fedora 9-14, RHEL 6), нет обычной команды, которая предотвращала бы автозапуск. Соответствующий конфигурационный файл Upstart из каталога `/etc/init` нужно изменить в текстовом редакторе либо деинсталлировать пакет.

8.6. Автоматический запуск процессов (cron)

Если ваш компьютер вдруг, как кажется, без причины, начнет производить поиск по диску, присылать вам почту и т. д., то, скорее всего, это работа демона `cron`, который автоматически запускает процессы. Демон активируется раз в минуту, анализирует все файлы `crontab` и запускает указанные в них программы. Он применяется, в первую очередь, в ходе работ по поддержанию системы — для удаления устаревших файлов регистрации и временных файлов, для обновления каталогов, создания резервных копий и т. д.

СОВЕТ

При обычной установке Linux программа Cron инсталлируется автоматически. Но учитывайте, что при минималистичных вариантах установки, которые часто применяются при работе с сервером или при виртуализации, Cron часто отсутствует. В таком случае выполните yum install vixie-cron (RHEL) или apt-get install cron (Debian/Ubuntu).

Глобальная конфигурация `cron` производится в файле `/etc/crontab`. Кроме того, пользователи могут задавать собственные задачи для демона `cron`, определяя их в каталоге `/var/spool/cron/tabs/user`.

Права на управление пользовательскими демонами `cron` регулируются в файлах `/var/spool/cron/allow` и `/deny`. Если файл `allow` существует, то команды `cron` могут выполнять только те пользователи, которые указаны в этом файле. Если существует файл `/deny`, то указанные в нем пользователи лишены такого права. Если оба файла отсутствуют, выполнение задач зависит от компиляции демона `cron`, в частности то, смогут ли им пользоваться какие-либо пользователи, кроме администратора.

Файл crontab

Файл `/etc/crontab` или файлы в каталоге `/etc/cron.d` содержат построчные записи, в которых перечисляются программы, предназначенные для выполнения (табл. 8.3). Синтаксис таков:

```
# in /etc/crontab
минута час день месяц неделя пользователь команда
```

Таблица 8.3. Столбцы crontab

Столбец	Значение
Минута	Указывает, в какую минуту часа (0–59) должна выполняться программа
Час	Указывает час (0–23)
День	Указывает день месяца (1–31)
Месяц	Указывает месяц (1–12)
Неделя	Указывает день недели (0–7, и 0, и 7 соответствуют воскресенью)
Пользователь	Указывает, какой пользователь должен выполнять команду (обычно это администратор)
Команда	Получает команду, которую необходимо выполнить

Если в любом из первых пяти полей вместо числа стоит символ *, то это поле игнорируется. Например, `15****` означает, что команда должна выполняться через каждые 15 минут после истечения очередного часа и так каждый час, каждый день, каждый месяц, независимо от дня недели. Запись `29 0 ** 6` означает, что команда должна выполняться каждую субботу в 0:29.

В тех полях, где обозначается время, также допускается запись вида `*/`*n*. Это означает, что команда должна выполняться в каждую *n*-ю минуту (час). Например, `*/15****` говорит о том, что команда должна выполняться через каждые 15 минут (*n*:00, *n*:15, *n*:30 и *n*:45). К сожалению, документация по этому демону еще не доработана, и из нее не вполне ясно, что означает `0` применительно ко дню месяца или месяцу года (см. `man 5 crontab`).

Файлы `/var/spool/cron/tabs/user` имеют тот же формат, что и `crontab`. Единственное отличие — отсутствует столбец *пользователь*.

Чтобы изменить глобальную конфигурацию `cron`, можно обработать файл `/etc/crontab` или файлы в каталоге `/etc/cron*` прямо в текстовом редакторе. Если к тому же вы хотите внести в `cron` специфические пользовательские дополнения, для этого следует использовать команду `crontab -e` (сначала выполните `export EDITOR=emacs`, если не хотите работать с vi; более подробная информация содержится на справочных сайтах, посвященных `cron` и `crontab`).

Каталоги cron.hourly, .daily, .weekly, .monthly

В большинстве дистрибутивов по умолчанию задается такая конфигурация, что `/etc/crontab` должен содержать всего несколько записей, необходимых для ежечасного выполнения сценариев, которые в свою очередь расположены в `/etc/cron.hourly/*`, для ежедневного выполнения — сценариев из `/etc/cron.daily/*` и т. д. в Ubuntu каталог `/etc/crontab` выглядит так:

```
# /etc/crontab
SHELL=/bin/sh
PATH=/usr/local/sbin:/usr/local/bin:/sbin:/bin:/usr/sbin:/usr/bin
17 *    * * *   root    cd / && run-parts --report /etc/cron.hourly
25 6    * * *   root    test -x /usr/sbin/anacron || \
                 ( cd / && run-parts --report /etc/cron.daily )
47 6    * * 7   root    test -x /usr/sbin/anacron || \
                 ( cd / && run-parts --report /etc/cron.weekly )
52 6    1 * *   root    test -x /usr/sbin/anacron || \
                 ( cd / && run-parts --report /etc/cron.monthly )
```

Это означает, что демон `cron`:

- через 17 минут после истечения каждого часа выполняет все сценарные файлы, находящиеся в каталоге `/etc/cron.hourly`;
- ежедневно в 6:25 выполняет все сценарные файлы из каталога `/etc/cron.daily`;
- еженедельно в 6:47 в воскресенье выполняет все сценарные файлы из каталога `/etc/cron.weekly`;
- в первое число каждого месяца в 6:52 выполняет все сценарные файлы из каталога `/etc/cron.monthly`.

Если установлена программа Anacron, то сценарии из /etc/cron.daily, -.weekly и -.monthly выполняться не будут.

Чтобы самостоятельно регулярно выполнять сценарии, занимающиеся резервным копированием, обновлениями и другими задачами, просто поместите соответствующий сценарный файл в один из каталогов: /etc/cron.hourly, -.daily, -.weekly или -.monthly. Не забудьте установить бит execute (chmod a+x *файл*)!

ВНИМАНИЕ

Имена файлов собственных сценариев cron.daily, cron.weeky и cron.monthly могут состоять из чисел, букв, дефисов и нижних подчеркиваний. Если в имени будет содержаться хотя бы одна точка, run-parts проигнорирует такой сценарий!

Если вы хотите дополнительно свериться с тем, какие сценарии из run.daily должны выполняться ежедневно, выполните команду run-parts --test /etc/cron.daily.

Если необходимо изменить время, заданное в crontab для работы с каталогами /etc/cron.*xxx*, то, разумеется, можете вписать в /etc/crontab дополнительную строку и указать время, когда должен выполняться ваш сценарий. Еще нагляднее было бы создать в /etc/cron.d соответствующий новый файл. Обратите внимание, что все конфигурационные файлы cron должны заканчиваться разрывом строки, иначе система будет игнорировать последнюю строку!

Планировщик задач Anacron

Наряду с демоном cron в большинстве дистрибутивов также установлен планировщик задач Anacron. Задача Anacron — обеспечивать ежедневное, еженедельное или ежемесячное выполнение различных задач также и в том случае, когда компьютер используется с перерывами, а по ночам и в выходные бывает выключен.

Anacron однократно (по требованию) выполняет сценарии из каталогов /etc/cron.daily, -.weekly и -.monthly. в отличие от cron, он завершает работу после выполнения необходимых сценариев. Anacron не отвечает за выполнение задач из разряда hourly — за них в любом случае отвечает исключительно cron.

ВНИМАНИЕ

На сервере, который может неделями работать без перерывов, применять Anacron нецелесообразно. Anacron выполняет задачи cron в каталогах /etc/cron.daily, -.weekly и -.monthly только по одному разу, а cron, в свою очередь, игнорирует эти задачи, если на компьютере установлен Anacron. в результате все без исключения задачи будут выполнены только по одному разу, независимо от того, как долго работает сервер!

Anacron сохраняет момент выполнения задачи в файлах каталога /var/spool/anacron. Таким образом, исключается ситуация, в которой определенный сценарий выполнялся бы дважды в один и тот же день. Anacron управляется из каталога /etc/anacrontab; конфигурация, задаваемая по умолчанию, обычно вполне подходит.

9 Конвертер графических, аудио- и текстовых файлов

В Linux имеется множество команд, с помощью которых можно преобразовывать изображения, тексты и другие файлы из одних форматов в другие: из GIF в JPEG, из PostScript в PDF, из HTML в обычный текст, из MP3 в WAV и т. д.

В этой главе я приведу краткий обзор таких команд и покажу примеры их использования. Если в вашей системе не окажется каких-либо команд, рассмотренных в данной главе, найдите и установите соответствующий пакет (однако обычно таких ситуаций не бывает).

9.1. Графический конвертер

Среди многих графических конвертеров, разработанных для Linux, выделяются два: Image Magick и Netpbm. Оба пакета приспособлены к работе с множеством графических форматов, содержат команды (параметры), упрощающие работу с изображениями (изменение изображения, его фрагмента, улучшение контрастности, уменьшение количества цветов и т. д.). Далее я коротко представлю оба пакета, а также некоторые входящие в их состав команды и библиотеки, применяемые для преобразования файлов.

Image Magick. Программный пакет Image Magick состоит из множества отдельных команд, самая важная из которых — `convert`. Она создает из имеющегося файла изображения новый файл, изменяя при этом формат. Исходный и конечный форматы понятны уже по имени файла. Например, следующая команда создает файл `image.png` в формате PNG:

```
user$ convert bild.jpg bild.png
```

Благодаря наличию более 100 параметров можно одновременно изменять различные свойства изображения (размер, количество цветов, степень сжатия и т. д.):

```
user$ convert -resize 100x100 bild.jpg bild.png
user$ convert -type Grayscale bild.jpg bild.eps
user$ convert -quality 80 bild.bmp bild.jpg
```

Команда mogrify работает подобно convert, но изменяет существующий файл (не создавая нового):

```
user$ mogrify -resize 50% test.jpg
```

Наконец, кратко рассмотрим еще некоторые команды: compare сравнивает два изображения, conjure выполняет команды сценарного языка Magick Scripting Language (MSL). Команда identify возвращает описание файла изображения (формат, размер и т. д.), import создает скриншот и сохраняет изображение в файле, montage делает из нескольких изображений новую картинку.

```
user$ identify -verbose bild.png
Image: bild.png
  Format: PNG (Portable Network Graphics)
  Geometry: 85x100
  Type: TrueColor
  ...
```

Пример небольшого shell-сценария, создающего эскизы (миниатюрные варианты) всех изображений, файлы с которыми были переданы сценарию в виде параметров, приводится в подразделе «Пример 4: сценарий резервного копирования» раздела 6.8. Подробная документация по всем командам находится на сайте http://www.imagemagick.org/.

Если вы хотите работать с функциями Magick в программе с графическим пользовательским интерфейсом, предлагаю познакомиться с программой Converseen: http://converseen.sourceforge.net/.

Netpbm. По принципу, напоминающему работу Image Magick, функционирует пакет Netpbm (ранее — Portable Bitmap Utilites). Для обработки в данном пакете любые файлы необходимо преобразовывать во внутренний формат Pnm или Pbm. в следующем примере показано преобразование файла из формата TIFF в формат EMS с одновременной нормализацией цветового формата изображения (pnmnorm):

```
user$ tifftopnm bild.tif | pnmnorm | pnmtops -noturn -rle -scale 0.5 > bild.eps
```

На сайте http://netpbm.sourceforge.net/doc/ приводится описание около 200 команд пакета Netpbm.

Библиотека libtiff. Пакет libtiff содержит одноименную библиотеку, а также различные команды для обработки и преобразования TIFF-файлов. Важнейшие команды для преобразования изображений — bmp2tiff, gif2tiff, tiff2pdf и tiff2ps. При работе с TIFF-файлами будут особенно полезны команды tiffcp, tiffinfo и tiffsplit.

Библиотека libwmf. Пакет libwmf содержит одноименную библиотеку, а также различные команды для обработки и преобразования WMF- и EMF-файлов (метафайлы среды Windows или улучшенные метафайлы). Важнейшие команды для преобразования изображений — wmf2eps, wmf2svg и wmf2gd (преобразование в форматы JPEG и PNG).

SVG-конвертер. В зависимости от дистрибутива пакет librsvg2 или librsvg2-bin может содержать команды rsvg и rsvgconvert, позволяющие преобразовывать

файлы в формате SVG (масштабируемая векторная графика) в точечные изображения.

EXIF. В некоторых дистрибутивах имеются различные библиотеки и команды для преобразования EXIF-файлов в JPEG, например `exif`, `exiftran` или `exiv2`.

RAW-файлы. Некоторые цифровые фотоаппараты позволяют сохранять снимки без потери качества в формате, заданном производителем (обычно патентованном). При преобразовании таких файлов в обычные изображения поможет команда `dcraw` из одноименного пакета.

9.2. Аудио- и видеоконвертер

В табл. 9.1 перечислены важнейшие команды, позволяющие считывать аудиофайлы с CD либо преобразовывать аудио- и видеофайлы из одного формата в другой. Если имя пакета не входит в название команды, то оно дополнительно указывается в скобках (название одного и того же пакета в различных дистрибутивах может варьироваться).

Таблица 9.1. Аудио- и видеоконвертер

Формат	Команда (пакет)
CD→WAV	icedax, cdparanoia
MP3→WAV	mpg123, mpg321, madplay
WAV→MP3	lame
OGG→WAV	oggdec (vorbis-tools)
WAV→OGG	oggenc (vorbis-tools)
MP3→OGG	mp32ogg
AAC→WAV	faad
WAV→AAC	faac
WAV↔FLAC	flac
Аудио↔аудио	sox
Аудио↔аудио	sfconvert (audiofile)
Аудио/видео↔аудио/видео	ffmpeg (общий аудио-, видеоконвертер)
Аудио/видео↔аудио/видео	mencoder (общий аудио-, видеоконвертер)

Далее я сообщу дополнительную информацию и приведу примеры по самым важным командам. Ради экономии места я не буду отдельно останавливаться на параметрах этих команд (если вам нужна дополнительная информация, то обратитесь к `man` *`имя_команды`*).

CD-риппер. В этом разделе мы поговорим, как лучше всего переписывать аудиотреки с компакт-диска на винчестер с максимальной эффективностью и минимальными потерями качества.

Среди наиболее популярных команд, которые разрабатывались специально для этого, необходимо отметить `cdparanoia` и `icedax` (ранее `cdda2wav`). Команда `cdparanoia` снискала отличную репутацию при работе с поцарапанными дисками и при решении других сходных проблем. Обе команды обладают большим количеством параметров.

Рассмотрим два примера. Первая команда считывает дорожку 3 с компакт-диска, который находится в приводе SCSI-CD. Полученный в результате файл называется audio.wav:

```
root# icedax -D /dev/scd0 -t 3
```

Следующая команда считывает дорожку 4 с компакт-диска, который находится в том же приводе. Полученный в результате файл называется cdda.wav и располагается в локальном каталоге:

```
root# cdparanoia -d /dev/scd0 "4"
```

MP3-кодер. Из-за проблем, связанных с патентами, сегодня практически не осталось дистрибутивов Linux, вместе с которыми поставляется MP3-кодер. Однако различные кодеры размещены в Интернете в виде дополнительных пакетов. Самая известная подобная программа называется lame (адрес ее сайта: http://lame.sourceforge.net/).

Работать с ней достаточно просто: команда lame input.wav output.mp3 создает из исходного файла в формате WAV соответствующий файл в формате MP3. Ход процесса и особенно желаемый уровень качества MP3-файла зависят от применения разнообразных параметров.

MP3→OGG. Перечисленные выше команды можно комбинировать, чтобы, например, преобразовать MP3-файл в формат Ogg-Vorbis. Однако следует учитывать, что при таких преобразованиях снижается качество материала, поэтому их по возможности следует избегать.

```
user$ mpg321 -s in.mp3 -w - | oggenc - -o out.ogg
```

К сожалению, при преобразовании также теряются информационные метки (ID3). Однако от этого недостатка можно избавиться, применяя сценарий MP3-Ogg-Konverter (mp32ogg с сайта http://faceprint.com/code/).

FLAC. FLAC означает Free Lossles Audio Codec (дословно «свободный аудиокодек без потерь»). Файлы FLAC несколько больше, чем MP3- или OGG-файлы, но они гораздо меньше WAV-файлов. Значительное преимущество по сравнению с MP3 или OGG заключается в том, что с помощью формата FLAC можно кодировать аудиофайлы без потерь. Для кодирования и декодирования используется команда flac.

SoX. SoX означает Sound Exchange, а команда sox обеспечивает еще одну возможность преобразования файлов из одного формата в другой. Она поддерживает больше форматов, чем sfconvert. Для sox существует X-интерфейс xsox, а также Gnome-интерфейс gsox.

Библиотека Audio File. Пакет audiofile внедряет важнейшие функции одноименной библиотеки, разработанной производителем SGI. Самая интересная команда — sfconvert: она конвертирует форматы аудиофайлов AIFF, AIFC, NEXT и WAVE. Команда sfinfo пытается установить, в каком формате записан аудиофайл.

SoundConverter. Если вы хотите конвертировать аудиофайлы с большим удобством, вам, вероятно, понравится программа SoundConverter. Она преобразует все выбранные заранее аудиофайлы в желаемый формат (меню Правка ▸ Настройки, по умолчанию задан формат Ogg-Vorbis). Новые файлы сохраняются в том же ката-

логе, что и исходные, однако, конечно же, получают новое расширение (например, *.ogg).

Видеоконвертер (ffmpeg). Команда ffmpeg из одноименного пакета преобразует аудио- и видеофайлы из одного формата в другой. Список поддерживаемых форматов очень велик, его можно узнать с помощью команды ffmpeg -formats. При настройке параметров учитывайте, что они действуют для файла, заданного в качестве следующего. При этом для правильной работы команды решающее значение имеет порядок следования параметров. Если не сделать никаких противоречащих этому настроек, ffmpeg применяет при создании конечного файла те же кодеки и настройки, которые использовались при создании исходного файла и таким образом позволяет избежать снижения качества. В следующем примере ffmpeg создает файл с фильмом в DVD-формате:

```
user$ ffmpeg -i in.avi out.mpg
user$ ffmpeg -i in.avi -y -target pal-dvd out.avi
```

Кроме того, ffmpeg подходит для записи аудио- и видеофайлов на DVD. Соответствующий пример приводится в man ffmpeg. Если вы не хотите запоминать многочисленные настройки ffmpeg, то можете воспользоваться для преобразования видеофайлов программой с графическим интерфейсом, например ffmpeg-GUI или winff.

Альтернативой для ffmpeg является команда mencoder. Она применяет ту же базу кода, что и видеоплеер MPlayer, и устанавливается вместе с ним. Mencoder отлично распознает все аудио- и видеоформаты, поддерживаемые MPlayer.

9.3. Текстовые конвертеры (кодировка и разрывы строк)

В этом разделе будут представлены команды recode, iconv, unix2dos и dos2unix. Они применяются для изменения кодировки и символов разрыва строки в обычных текстовых файлах. Это требуется в тех случаях, когда текстовые файлы передаются между системами, в которых отличаются кодировки или конвенции текстовых форматов.

Команда recode. Преобразует набор символов из кодировки 1 в кодировку 2. Следующая команда преобразует DOS-файл dosdat в файл с кодировкой «латиница-1»:

```
user$ recode ibmpc..latin1 < dosdat > linuxdat
```

Команда recode считывает текстовый файл latin1dat с кодировкой «латиница-1» и сохраняет его в формате UTF-8 (Unicode):

```
user$ recode latin1..u8 < latin1dat > utf8dat
```

Команда iconv. Популярная альтернатива для recode — команда iconv. Правда, эта команда не может изменять символы разрыва строки. В следующем примере создается файл UTF-8 из соответствующего файла в кодировке «латиница-1»:

```
user$ iconv -f latin1 -t utf-8 latin1dat > utf8dat
```

Команды dos2unix, unix2dos. Эти команды изменяют символы разрыва строки при преобразовании файла из формата, типичного для MS-DOS/Windows (CR плюс LF) в формат, типичный для UNIX/Linux (только LF). Команды пригодны для работы с такими текстовыми файлами, которые содержат только однобайтовые последовательности символов (например, ASCII, «латиница-1»), но не для файлов в формате Unicode!

```
user$ dos2unix файл.txt
```

9.4. Конвертер имен файлов (кодировка)

Команда convmv. Еще несколько лет назад во многих дистрибутивах Linux было принято записывать имена файлов в «латинице-1». Однако со временем стандартной кодировкой стал считаться Unicode (UTF-8). При переводе имен файлов из одной кодировки в другую используется команда `convmv`, однако она редко имеется в дистрибутиве по умолчанию. В некоторых дистрибутивах ее можно без труда установить в виде одноименного пакета. Если же для конкретного дистрибутива не предусмотрен соответствующий пакет, то вам потребуется скачать сценарий на Perl с сайта http://j3e.de/linux/convmv/.

Чтобы преобразовать все файлы одного каталога из кодировки «латиница-1» в UTF-8 (в обратном порядке, с запросом о подтверждении преобразования каждого отдельно взятого файла), вызовите `convmv` следующим образом:

```
user$ convmv -r -i --notest -f iso-8859-1 -t utf8 название_каталога
```

Команда `convmv` изменяет только имена, а не содержимое файлов! При первых испытаниях рекомендую отказаться от параметра `--notest`: тогда `convmv` только отобразит запланированные изменения, но не будет их выполнять.

Команда `convmv` пытается сама опознавать имена файлов, которые уже работают с кодировкой UTF-8, и в таком случае не изменяет имени файла. Подобный механизм защиты отключается с помощью параметра `--nosmart`.

9.5. Конвертер документов (PostScript, PDF, HTML, LATEX)

В этом разделе представлены команды, применяемые при обработке и преобразовании документов, созданных в форматах PostScript, PDF, HTML и т. д. В табл. 9.2 представлен их список.

Таблица 9.2. Конвертеры документов

Формат	Команда
Text→PostScript	a2ps, enscript, mpage
HTML→Text, PostScript	html2text, html2ps
PostScript→PDF	ps2pdf, epstopdf, pdf2ps, pdftops
PostScript, PDF→Bitmap, Druckerformat	gs

Формат	Команда
PostScript→PostScript (извлечение сайтов и т. д.)	psutils
PDF→PDF (извлечение рисунков/сайтов и т. д.)	pdftk, pdfnup, pdfjoin, pdfedit
PDF→Text	pdftotext
LATEX→DVI, PostScript, PDF	latex, dvips, dvipdf, dvipdfm

Text→PostScript

Если вы распечатываете текстовые файлы с помощью команды `lpr` *файл*, то система печати обычно автоматически форматирует текст. Если у вас есть свои соображения относительно того, как должен быть отформатирован полученный фрагмент текста, то рекомендуется отформатировать файл PostScript вручную и дополнительно его распечатать. Для выполнения этой задачи подходят, в частности, команды `a2ps`, `enscript` и `mpage`. Функционал этих трех команд одинаков, они отличаются только предлагаемым набором параметров форматирования.

Команда a2ps. Эта команда расшифровывается как Any to (Что-то в PostScript) и может преобразовывать в PostScript, например, текстовые файлы справки. В следующих примерах мы ограничимся обычными текстовыми файлами. Обратите внимание на то, что в текстовых файлах должна использоваться кодировка-латиница (а не Unicode)!

По умолчанию команда форматирует текст в виде двух столбцов с альбомной ориентацией.

```
user$ a2ps text.txt -o postscript.ps
```

Следующая команда обрабатывает несколько текстовых файлов и форматирует полученный результат как «четыре страницы на листе». Если `a2ps` опознает текст как программный код, она автоматически выделит синтаксис (ключевые слова — полужирным шрифтом, комментарии — курсивом и т. д.).

```
user$ a2ps файл1.c файл2.c файл3.h -4 -o postscript.ps
```

Команда enscript. Преобразует текстовые файлы в форматы PostScript, HTML и RTF. Она работает с текстовыми файлами в кодировке «латиница-1». Следующая команда создает страницы формата A4 с альбомной ориентацией и тремя столбцами текста:

```
user$ enscript -M A4 --landscape -3 text.txt -p postscript.ps
```

Команда mpage. Эта команда также преобразует текстовые файлы в формат PostScript, по умолчанию по четыре страницы на лист в формате «письма». Следующая команда создает страницы A4 с альбомной ориентацией, с двумя страницами на листе:

```
user$ mpage -2 -bA4 text.txt > postscript.ps
```

Unicode. К сожалению, ни одна из указанных выше программ не может работать с кодировкой Unicode (UTF-8). Если вы хотите распечатать документы с кодировкой Unicode, то лучше всего воспользоваться редактором, поддерживающим ее.

Чтобы просто и быстро преобразовать текст в PostScript, можно также применить команду `cnprint`: http://www.neurophys.wisc.edu/~cai/software.

HTML→Text, PostScript

Команда html2text. Преобразует HTML-документы в обычные текстовые файлы. Это бывает полезно, когда HTML-файлы необходимо передать в виде, удобном для чтения в браузере.

```
user$ html2text файл.html > text.txt
```

Команда `html2text` выдает текст в кодировке «латиница-1» (не Unicode!). Форматирование текста управляется несколькими параметрами, а также `/etc/html2textrc` или `~/.html2textrc` (см. `man html2textrc`).

Для преобразования HTML в текст можно также пользоваться браузерами с текстовой ориентацией, например, Lynx, ELinks или w3m.

Команда html2ps. Для автоматического преобразования данных из HTML в PostScript подходит сценарий на языке Perl — `html2ps`. Работать с ним достаточно просто: `html2ps -D имя.html > имя.ps`. Благодаря параметру `-D` в файл PostScript вставляются DSC-совместимые[1] комментарии, что сильно облегчает дальнейшую работу с файлом.

PostScript↔PDF

Команда ps2pdf. Команда `ps2pdf исходный.ps конечный.pdf` создает из любого файла PostScript файл формата PDF. Функционально эта программа, в принципе, не отличается от коммерческого инструмента Adobe Distiller. Она основана на Ghostscript (gs).

На настоящий момент команда создает файлы, совместимые с форматом PDF 1.2 для программы Acrobat Reader 3.*n*. Однако, судя по документации, в будущем установки, задаваемые по умолчанию, могут измениться. Если вы хотите гарантировать совместимость с конкретной версией PDF, используйте команды `ps2pdf12`, `ps2pdf13` и `ps2pdf14`.

Качество файлов в формате PDF во многом зависит от того, какие шрифты используются в документе PostScript. Если определенные шрифты не поддерживаются, то их символы необходимо заменять точечными рисунками, из-за чего качество шрифта серьезно снижается. Работа `ps2pdf` регулируется многочисленными параметрами. Полная документация находится по адресу http://www.cs.wisc.edu/~ghost/doc/cvs/Ps2pdf.htm.

Команда epstopdf. Если вы хотите преобразовать изображение EPS в файл PDF, воспользуйтесь командой `epstopdf`. EPS расшифровывается как Encapsulated PostScript («инкапсулированный PostScript») и включает в себя такие файлы, в которых с помощью так называемой *граничной рамки* задается точный размер рисунка. Файлы в формате EPS хорошо встраиваются в другие документы (например, LATEX или OpenOffice).

---

[1] DSC — Соглашения о структурировании документа (Document Structuring Conventions).

Команда `epstopdf` обычно входит в состав пакета `tetex`, содержащего TEX, LATEX и другие TEX-специфичные программы. В отличие от `ps2pdf`, команда `epstopdf` учитывает размер изображения. К сожалению, `epstopdf` не может переносить в PDF точечные рисунки, содержащиеся в файле EPS, в неизменном виде (а также файлы с параметром `--nocompress`).

Команды pdf2ps и pdftops. Команда `pdf2ps` *`исходный.pdf конечный.ps`* — это команда, обратная `ps2pdf`; `pdf2ps` также работает с командой `gs`.

Команда `pdftops` выполняет, в принципе, те же задачи, что и `pdf2ps`, но иначе встраивается в систему и содержит значительно больше параметров, влияющих на вид итоговых файлов PostScript. Например, можно указывать желаемый уровень PostScript, размер бумаги и т. д.

PostScript/PDF→формат для вывода на печать/точечная графика

Команда gs. Эта команда, также известная под названием Ghostscript, преобразует документы PostScript и PDF в различные форматы для вывода на печать и с использованием точечных рисунков. Ghostscript — это важнейший элемент системы печати в Linux (например, CUPS), так как он обеспечивает распечатку документов в формате PostScript на принтерах, не поддерживающих этот формат. Кроме того, эта программа применяется в различных инструментах для просмотра и преобразования PostScript.

Команда `gs` использует шрифты, установленные на компьютере, а также собственную коллекцию гарнитур, которая обычно находится в пакете `ghostscript-fonts`. Эти гарнитуры необходимы для того, чтобы преобразовать шрифты PostScript в точечную графику.

Существуют различные версии Ghostscript. В большинстве дистрибутивов Linux применяется GNU Ghostscript или его вариант ESP Ghostscript. Обе версии соблюдают лицензию GPL. Вариант ESP Ghostscript специально оптимизирован под совместную работу с CUPS. ESP — это название компании, означает Easy Software Products.

Кроме того, имеются коммерческие версии Ghostscript, например Artifex Ghostscript, продаваемые, например, производителями принтеров. Более подробная информация по различным версиям имеется на следующих сайтах:

- http://www.ghostscript.com/;
- http://www.artifex.com/.

Внешние драйверы принтеров (Gutenprint). В Ghostscript интегрировано множество драйверов для принтеров. Однако многие из них были разработаны вне проекта Ghostscript. Важнейший современный проект по разработке драйверов называется Gutenprint (ранее — Gimp-Print). Более подробная информация находится по адресу http://gimp-print.sourceforge.net/.

Здесь также стоит упомянуть о проекте HPLIP (HP Linux Imaging and Printing). В рамках этого проекта компания Hewlett-Packard предоставляет свободно распространяемые драйверы для различных принтеров и сканеров. Однако HPLIP не связан с Ghostscript и применяется в сочетании с CUPS.

Вызов вручную. Поскольку Ghostscript хорошо интегрирован в печатную систему и многие другие программы, вызывать gs вручную приходится нечасто. Чтобы gs работала правильно, необходимо задать как минимум два параметра: -sOutputFile= для указания файла, в который заносится результат, и -sDEVICE=*имя* или @*имя*.upp для настройки формата вывода. Как правило, стоит также использовать параметр -dNOPAUSE. Если вы собираетесь печатать на бумаге DIN-A4, то укажите еще и -sPAPERSIZE=a4.

Следующая команда переводит test.ps в формат принтера HP-Laserjet 3. Результат записывается в файл out.hp:

```
user$ gs -sDEVICE=ljet3 -sOutputFile=out.hp -sPAPERSIZE=a4 -dNOPAUSE -dBATCH test.ps
```

Во втором примере PostScript преобразуется в формат PDF (ps2pdf — это просто сценарий, вызывающий gs):

```
user$ gs -dNOPAUSE -dBATCH -sDEVICE=pdfwrite -sOutputFile=out.pdf test.ps
```

И, наконец, вот команда, преобразующая файл EPS в формат PNG:

```
user$ gs -dNOPAUSE -dBATCH -sDEVICE=png16m -sOutputFile=out.png \
        -dEPSCrop -r100 bild.eps
```

Утилиты PostScript

Пакет psutils. При обработке файлов PostScript очень полезны команды из пакета psutils (табл. 9.3). Часть из них — самостоятельные программы, часть — сценарии bash или Perl.

Таблица 9.3. Команды пакета psutils

Команда	Функция
epsffit	Регулирует размер EPS-файла
extractres	Анализирует файл и вставляет комментарии %%IncludeResource для всех необходимых шрифтов, файлов и т. д.
fixfmps	Приспосабливает файлы FrameMaker к правилам psutils
fixmacps	Приспосабливает файлы Macintosh к правилам psutils
fixscribeps	Приспосабливает файлы Scribe к правилам psutils
fixtpps	Приспосабливает файлы Troff/Tpscript к правилам psutils
fixwfwps	Приспосабливает файлы Microsoft Word к правилам psutils
fixwpps	Приспосабливает файлы WordPerfect к правилам psutils
fixwwps	Приспосабливает файлы Microsoft Write к правилам psutils
getafm	Создает файлы AFM для описания шрифтов
includeres	Вставляет комментарии, созданные extractres, в файл PostScript
psbook	Упорядочивает страницы с текстом так, что их можно печатать в виде тетрадей по 16 листов
psnup	Размещает на одном листе несколько уменьшенных страниц
psresize	Изменяет параметры страницы в документе. Эта команда позволяет решить проблему, регулярно возникающую при распечатке документов PostScript, созданных в формате «письма», который используется в США

Команда	Функция
psselect	Извлекает отдельные страницы из файла PostScript
pstops	Переупорядочивает страницы в документе

В следующем примере показано, как преобразовать файл PostScript с рукописью этой книги, созданный с применением LATEX и DVIPS в форму «64 страницы на листе». Таким образом, каждая страница уменьшится до размеров почтовой марки. При работе с программой для просмотра открываются возможности быстрого просмотра внешнего вида страниц (как в режиме Предварительный просмотр в Microsoft Word с максимальным уменьшением):

```
user$ psnup -b-0.4cm -64 -q < linux.ps > preview.ps
```

К сожалению, вышеуказанные команды функционируют лишь тогда, когда файлы PostScript содержат DSC-совместимые комментарии (как было указано выше, DSC означает «Соглашения о структурировании документа»). Комментарии не распечатываются, но содержат важную информацию о размере страницы, ее начале, конце и т. д. Файлы EPS — это одностраничные файлы PostScript, включающие в себя комментарии о том, как вставлять этот материал в другие документы (в частности, данные о граничной рамке, определяющей размер распечатки).

Объединение файлов PS. Чтобы объединить два и более файлов PostScript, нужно воспользоваться командой Ghostscript `gs`. Правда, достичь удовлетворительных результатов удастся лишь в том случае, если она найдет все файлы шрифтов.

```
user$ gs -sDEVICE=pswrite -sOutputFile=out.ps -dNOPAUSE -dBATCH in1.ps in2.ps ...
```

Утилиты PDF

Команда pdftk. Эта команда (инструментарий PDF) содержит функции для обработки PDF, подобные тем, что `psutils` предлагает для файлов PostScript. С помощью этого инструментария можно извлекать страницы, объединять несколько PDF-документов, создавать незашифрованную версию зашифрованного документа PDF (при условии, что вы знаете пароль), заполнять PDF-формы и т. д. Подробная информация по синтаксису инструментария приводится на сайте http://www.accesspdf.com/pdftk/.

Следующая команда считывает страницы 10–20 и 30–40 из файла `in.pdf` и записывает их в новый файл `out.pdf`:

```
user$ pdftk in.pdf cat 10-20 30-40 output out.pdf
```

Чтобы объединить несколько файлов в формате PDF, используйте команду `cat`:

```
user$ pdftk in1.pdf in2.pdf in3.pdf cat output out.pdf
```

В следующем примере для каждой отдельной страницы из файла `in.pdf` создается отдельный файл PDF с именем `pg_`*n*, где *n* — номер страницы. Если вы хотите дать файлу другое имя, то нужно передать `output` последовательность символов (строку), соответствующую синтаксису `printf`, например `output page-%02d.pdf`:

```
user$ pdftk in.pdf burst
```

В следующем примере создается зашифрованный PDF-файл. Его можно читать и без пароля xxx, но распечатывать или вносить изменения нельзя. Если вы хотите защитить файл даже от чтения, используйте вместо owner_pw команду user_pw.

```
user$ pdftk in.pdf output encrpyted.pdf owner_pw xxx
```

Poppler. Это собрание команд, предназначенных для преобразования PDF-документов в другие форматы (Text, Bitmap, PostScript и т. д.). Poppler в Linux используется во многих программах, предназначенных для просмотра PDF. Обычно эта программа находится в пакете poppler-utils.

Пакет xpdf-utils. Этот пакет содержит среди прочих команды pdftops (преобразует документы формата PDF в PostScript), pdfinfo (извлекает свойства документов PDF), pdfimages (извлекает изображения из файлов PDF) и pdftotext (извлекает текст из файлов PDF).

Пакет pdfedit. Включает в себя различные инструменты и пользовательский интерфейс, предназначенные для изменения файлов PDF.

Пакет pdfjam. Содержит команды pdfnup, pdfjoin и pdf90. Они могут объединять и поворачивать документы PDF.

GUI. Если же вам не нравится работать с командами и их параметрами и не хватает графического пользовательского интерфейса, то вам предлагается богатый выбор свободно распространяемых программ, часто основанных на Java. Кроме уже упомянутой программы PDFedit, достаточно интересны, прежде всего, PDF-Shuffler, Bookbinder, JPDF Tweak, а также PDF Split и Merge (PDF Sam).

LATEX и компания

LATEX — это система для набора (верстки) научных текстов. В данном подразделе будут кратко описаны важнейшие команды, позволяющие преобразовывать файлы LATEX (*.tex) в другие форматы, но вдаваться в подробности синтаксиса LATEX мы не будем.

Команда latex. Команда latex *имя*.tex создает из LATEX файл в формате DVI. Он содержит все команды, необходимые для верстки страницы на принтеро- или устройство-независимом языке.

Команда dvips. Когда файл DVI будет готов, его можно просмотреть с помощью программ xdvi или kdvi. Команда dvips позволяет преобразовать файл DVI в формат PostScript. На примере следующей команды понятен принцип построения синтаксиса в таких случаях.

```
user$ dvips [параметры] -o имя.dvi имя.ps
```

Команда dvipdf. Часто бывает необходимо преобразовать документы LATEX в PDF. Для этого имеется несколько возможностей.

- Сначала с помощью dvips создается файл PostScript, который затем преобразуется в PDF с помощью ps2pdf или Adobe Distiller (Adobe Distiller входит в состав коммерческого программного пакета Adobe Acrobat, однако, к сожалению, пока не существует версии этой программы для Linux).
- Можно преобразовать файл DVI в PDF с помощью dvipdf или dvipdfm. При этом dvipdf соответствует пункту, описанному выше, так как в качестве промежуточного этапа тоже создается файл PostScript.

В результате имеем файл PDF, выглядящий так же, как и эквивалентный файл PostScript. Можно ли будет использовать дополнительные функции PDF (интерактивное оглавление, переходы по ссылкам), зависит от способа преобразования и дополнительных пакетов, применявшихся при создании документа LATEX:

- `dvips/ps2pdf` или `dvipdf` — функции PDF могут использоваться с помощью пакета LATEX hyperref;
- `dvipdfm` — в данном случае следует вставить дополнительные команды `\special` в документ LATEX;
- `PDFTEx` — в этой программе имеются дополнительные команды LATEX, предназначенные для управления функциями PDF.

10 Сетевые инструменты

В этой главе будут представлены команды, предназначенные для работы с простейшими сетевыми службами. Здесь мы рассмотрим, как с использованием `ssh` войти в сеть с другого компьютера, как передать файлы с помощью команды `wget` или `rsync` и т. д.

10.1. Определение состояния сети

В данном разделе приводится обзор команд, применяемых для тестирования основных функций сети. Более подробная информация по упоминаемым здесь командам предоставлена в разделе 18.3, где мы рассмотрим, как конфигурировать доступ к сети вручную.

Определение сетевых интерфейсов

Команда `ifconfig` возвращает список всех известных сетевых интерфейсов. Обычно используются интерфейсы `eth`*n* (Ethernet) и `ppp`*n* (доступ к Интернету через модем, ADSL или VPN). Особую роль играет `lo`: этот интерфейс позволяет программам локальных компьютеров обмениваться информацией через сетевой протокол. Он работает даже тогда, когда соединение от компьютера к сети выполнено «наружу».

Обычно результат выглядит примерно так:

```
root#   ifconfig
eth0    Link encap:Ethernet Hardware Address 00:11:25:32:4F:5D
        inet Address:192.168.0.15 Bcast:192.168.0.255 Mask:255.255.255.0
        inet6 Address: fe80::211:25ff:fe32:4f5d/64 Scope:Connection
        UP BROADCAST RUNNING MULTICAST MTU:1500 Metric:1
        RX packets:11416 errors:0 dropped:0 overruns:0 frame:0
        TX packets:10415 errors:0 dropped:0 overruns:0 carrier:0
        Collisions:0 transmit queue:1000
        RX bytes:9600456 (9.1 MiB) TX bytes:2269502 (2.1 MiB)
        Base address:0x8000 memory:c0220000-c0240000
lo      Link encap:local loop
        inet Address:127.0.0.1 Mask:255.0.0.0
```

```
        inet6 Address: ::1/128 Scope:Machine
        ...
        RX bytes:6139586 (5.8 MiB) TX bytes:6139586 (5.8 MiB)
```

Если интерфейс отсутствует, это значит, что сетевая карта еще не активизирована. Вам поможет предусмотренный в дистрибутиве инструмент, предназначенный для конфигурирования сети. Вы также можете активизировать сетевой интерфейс вручную, например, так: `ifconfig eth0 192.168.0.2`. Если при этом возникает ошибка `eth0: unknown interface: No such device`, это означает, что в ядре отсутствует модуль для управления сетевой картой.

Тестирование доступности localhost

Программа `ping` ежесекундно посылает маленький сетевой пакет по указанному адресу. Если по этому адресу находится компьютер, он отправляет ответ (если брандмауэр не препятствует этому). Программа работает до тех пор, пока вы не завершите ее нажатием сочетания клавиш Ctrl+C. Локальный компьютер localhost проверяет наличие виртуального интерфейса для работы сетевых протоколов, который обеспечивает работу всех остальных функций компьютера.

```
user$ ping localhost
PING localhost (127.0.0.1): 56 data bytes
64 bytes from 127.0.0.1: icmp_seq=0 ttl=255 time=0.152 ms
64 bytes from 127.0.0.1: icmp_seq=1 ttl=255 time=0.114 ms
...
```

Тестирование доступности локальной сети

Передав в программу `ping` не localhost, а IP-адрес другого компьютера, находящегося в локальной сети, вы сможете проверить, как работает сеть. Команда `-c 2` гарантирует, что `ping` не будет работать бесконечно, а отошлет всего два пакета.

```
user$ ping -c 2 192.168.0.99
PING 192.168.0.99 (192.168.0.99): 56 data bytes
64 bytes from 192.168.0.99: icmp_seq=0 ttl=255 time=0.274 ms
64 bytes from 192.168.0.99: icmp_seq=1 ttl=255 time=0.150 ms
...
```

Если в локальной сети находится сервер имен, присваивающий имя IP-адресу 192.168.0.99 (или если эту задачу выполняет файл `/etc/hosts`), то можно указать `ping` не IP-адрес, а имя компьютера.

```
user$ ping -c 2 mars
PING mars.sol (192.168.0.10) 56(84) bytes of data.
64 bytes from mars.sol (192.168.0.10): icmp_seq=1 ttl=64 time=0.281 ms
64 bytes from mars.sol (192.168.0.10): icmp_seq=2 ttl=64 time=0.287 ms

--- mars.sol ping statistics ---
2 packets transmitted, 2 received, 0% packet loss, time 999ms
rtt min/avg/max/mdev = 0.281/0.284/0.287/0.003 ms
```

Тестирование доступа к Интернету

Потом можно проверить, как работает соединение с Интернетом. Следующая команда тестирует сразу два пункта сетевой конфигурации: доступность сервера имен и работу шлюзов.

```
user$ ping -c 2 www.yahoo.com
PING www.yahoo-ht2.akadns.net (209.73.186.238) 56(84) bytes of data.
64 bytes from f1.www.vip.re3.yahoo.com (209.73.186.238): icmp_seq=1 time=122 ms
64 bytes from f1.www.vip.re3.yahoo.com (209.73.186.238): icmp_seq=2 time=123 ms

--- www.yahoo-ht2.akadns.net ping statistics ---
2 packets transmitted, 2 received, 0% packet loss, time 999ms
rtt min/avg/max/mdev = 122.731/123.256/123.782/0.631 ms
```

Если это не работает, причин может быть несколько.

- Возможно, сервер Yahoo! сейчас недоступен либо на сервере из соображений безопасности была деактивирована функция ответа на `ping`. Попробуйте другой интернет-адрес.
- Если сервер имен не работает, вы получите сообщение об ошибке `ping: unknown host yahoo.com`. Проверьте, содержит ли файл `/etc/resolv.conf` адрес сервера имен.
- Если не работает шлюз, вы получите следующее сообщение об ошибке: `connect: Network is unreachable`. Выполните команду `route -n`. Последняя строка должна выглядеть примерно как в предыдущем примере, причем в столбце `Gateway` должен содержаться IP-адрес вашего шлюза:

```
root# route -n
Kernel IP routing table
Destination  Gateway       Genmask       Flags  Metric  Ref  Use  Iface
192.168.0.0  0.0.0.0       255.255.255.0 U      0       0    0    eth0
127.0.0.0    0.0.0.0       255.0.0.0     U      0       0    0    lo
0.0.0.0      192.168.0.10  0.0.0.0       UG     0       0    0    eth0
```

- Если функции шлюза выполняет один из компьютеров локальной сети, возможно, вы забыли о функции маскарадинга. В этом случае доступ к Интернету будет закрыт для всей сети. Более подробное руководство по конфигурации шлюза для доступа к Интернету дается в главе 19.

Отслеживание пути IP-пакетов

С помощью команды `traceroute` можно увидеть, какой путь проделал пакет с вашего компьютера на другой и сколько времени ушло на преодоление каждого отдельно взятого отрезка. По умолчанию команда предпринимает три попытки и возвращает три значения времени соответственно. Команда не работает, если на пути пакетов стоит брандмауэр, блокирующий порт UDP 33434, используемый `traceroute`. В таком случае команда возвращает для этого и всех последующих этапов только три звездочки.

Следующие строки показывают путь от моего рабочего компьютера к сайту google.at. Первая строка описывает интернет-шлюз (компьютер mars.sol), строка 2 — маршрутизатор ADSL, а строка 3 — шлюз интернет-провайдера. Начиная со строки 9, результат начинает дробиться между различными резервными компьютерами Google.

```
user$ traceroute google.at
traceroute to google.at (66.102.9.104), 30 hops max, 40 byte packets
 1 mars.sol.0.168.192.in-addr.arpa (192.168.0.1) 0.277 ms 0.194 ms 0.216 ms
 2 192.168.1.1 (192.168.1.1) 0.373 ms 0.367 ms 0.690 ms
 3 N704P030.adsl.highway.telekom.at (62.47.31.254) 8.598 ms 8.500 ms 11.593 ms
 4 172.19.90.193 (172.19.90.193) 11.864 ms 8.837 ms 7.986 ms
...
14 66.102.9.104 (66.102.9.104) 52.741 ms 53.306 ms 51.996 ms
```

Команда mtr. Команда `mtr` регулярно посылает сетевые пакеты указанному хосту и анализирует его отклики. В списке результатов комбинируются данные, получаемые от `ping` и `traceroute`. Обратите внимание, что существует две версии этой программы: описанный здесь текстовый вариант и GTK-вариант с графическим пользовательским интерфейсом. В настольных версиях Debian и Ubuntu по умолчанию устанавливается GTK-вариант. Чтобы установить вместо него текстовую версию, выполните `apt-get install mtr-tiny`.

```
user$ mtr -c 10 -r google.de
HOST: michaels-computer  Loss% Snt   Last  Avg   Best  Wrst  StDev
 1.|-- speedtouch.lan    0.0%  10    42.6  48.5  6.0   95.9  28.9
 2.|-- 178-191-207-254.adsl.high 0.0% 10    18.9  20.4  18.6  23.2  1.9
 3.|-- 195.3.74.129      0.0%  10    19.4  18.8  17.9  19.4  0.5
 4.|-- AUX10-GRAZBC10.highway.te 0.0% 10    21.2  21.3  20.7  22.0  0.3
 5.|-- 195.3.70.154      0.0%  10    21.2  27.3  20.9  81.2  18.9
 6.|-- 62.47.120.150     0.0%  10    25.3  25.5  24.9  26.0  0.4
 7.|-- 209.85.243.119    0.0%  10    25.6  25.9  25.2   28.2 0.8
 8.|-- 216.239.46.88     0.0%  10    25.8  26.3  25.8  27.7  0.6
 9.|-- bud01s08-in-f23.1e100.net 0.0% 10    25.7  25.8  25.0  26.8  0.5
```

Программа gnome-nettool. Если вы работаете с Gnome, то можете с удобством узнать большую часть вышеперечисленной информации с помощью программы `gnome-nettool`

10.2. Работа на других компьютерах (SSH)

Программы `telnet`, `rlogin` и `ssh` позволяют работать на другом компьютере, словно он стоит прямо перед вами. Этот принцип функционирует и в командно-ориентированных программах, и в X-программах. В этом разделе я ограничусь описанием `ssh` *(безопасного командного процессора)*. Устаревающими программами `telnet` и `rlogin` я не рекомендую пользоваться по соображениям безопасности, так как в них учетные данные, в том числе пароль, передаются в незашифрованном виде.

Для использования ssh прежде всего необходим второй компьютер, на котором будет работать сервер SSH, то есть программа sshd. В некоторых дистрибутивах Linux она установлена по умолчанию, в других ее требуется установить (как минимум пакет openssh-server). Если на всех компьютерах включены брандмауэры, они не должны блокировать порт 22.

ПРИМЕЧАНИЕ

Информация об установке, конфигурировании и обеспечении безопасности SSH-сервера дается в главе 22. Там вы также узнаете, как с достаточным удобством и безопасностью можно выполнять SSH-аутентификацию с ключами.

Обычное shell-соединение

Если вы работаете на компьютере uranus и хотите запустить shell-соединение на компьютере mars, то выполните следующую команду, чтобы установить соединение:

```
user@uranus$ ssh mars
user@mars's password: ********
```

Если вы впервые устанавливаете соединение с другим компьютером, то часто выводится предупреждение следующего вида:

```
The authenticity of host 'mars (192.168.0.10)' can't be established.
RSA1 key fingerprint is 1e:0e:15:ad:6f:64:88:60:ec:21:f1:4b:b7:68:f4:32.
Are you sure you want to continue connecting (yes/no)? {\bfs yes}
Warning: Permanently added 'mars,192.168.0.10' (RSA1) to the list
   of known hosts.
```

Это означает, что программа ssh не уверена, можно ли доверять компьютеру mars с IP-адресом 192.168.0.10. Возможно, чужой компьютер выдает себя за mars. Если ответить на запрос YES, то ssh сохранит имя, адрес и отличительную метку RSA (код для однозначной идентификации партнерского компьютера) в ~/.ssh/known_hosts.

Если вы хотите работать на компьютере mars под другим логином, нежели на uranus (например, в качестве администратора), укажите имя с параметром -l:

```
user@uranus$ ssh -l root mars
root@mars's password: xxx
```

СОВЕТ

Аутентификация с ключом гораздо удобнее, чем аутентификация с логином и паролем. Этот метод будет подробно описан в главе 22. При применении ключей вы также сможете автоматически выполнять команды, базирующиеся на SSH, и сценарии (например, сценарии резервного копирования).

Выполнение команд

Можно не использовать ssh в интерактивном режиме, а просто выполнить на удаленном компьютере какую-либо команду. Команда и ее параметры в таком случае будут переданы ssh в качестве дополнительных параметров. После этого ssh завершит работу.

```
user@uranus$ ssh mars команда параметры
user@mars's password: xxx
```

Эта, на первый взгляд, тривиальная функция открывает вам широкие возможности: теперь можно, например, запустить на удаленном компьютере `tar`, чтобы передать созданный архив стандартному выводу (ставим дефис после параметра `-f`), а затем с помощью | использовать стандартный вывод в качестве ввода для второй команды `tar`, работающей на локальном компьютере. Таким образом, можно безопасно копировать через SSH целые каталоги.

Следующая команда показывает, как я скопировал целое дерево каталогов `htdocs` с веб-сервера `kofler.cc` в локальный каталог `~/bak`:

```
user$ ssh -l username kofler.cc tar -cf - htdocs | tar -xC ~/bak/ -f -
username@kofler.cc's password: ******
```

SSH и X

В принципе соединение SSH позволяет выполнять и программу X. В таком случае программа запущена на удаленном компьютере, но ее работа отображается на локальном компьютере, откуда можно вводить информацию с помощью мыши и клавиатуры. Поскольку теперь весь X-протокол функционирует через сеть, вам понадобится хорошее сетевое соединение, с которым будет удобно работать.

Чтобы выполнить X-программу, ее нужно запустить с параметром `-X` (если `/etc/ssh_config` содержит строку `ForwardX11 yes`, то параметром `-X` можно пренебречь). Программа `ssh` сама правильно настроит переменные `DISPLAY`.

Следующие команды запускают на компьютере mars редактор XEmacs. Однако окно редактора открывается на экране компьютера uranus, откуда с редактором теперь можно работать! Этот механизм функционирует даже тогда, когда на компьютере mars не работает X-сервер (уровень запуска 3). Однако должны быть установлены все X-библиотеки!

```
user@uranus$ ssh -X mars
user@mars's password: xxx
user@mars$ xemacs &
```

Безопасное копирование файлов с помощью scp

Для копирования файлов через SSH по сети применяется команда `scp`. Ее синтаксис таков:

```
user$ scp [[user1@]host1:]имя_файла1 [[user2@]host2:][имя_файла2]
user2@host2's password: ******
```

Таким образом, файл *имя_файла1* копируется с компьютера host1 на компьютер host2, где сохраняется в файле *имя_файла2*. Некоторые замечания по опциональным составляющим команды копирования:

- указывать host1 и host2 не нужно, когда имеется в виду локальный компьютер (то есть localhost);
- не следует задавать user1, когда имеется в виду текущий пользователь;

- не нужно указывать user2, когда на компьютере host2 должно использоваться текущее имя пользователя с host1 или user1;
- *имя_файла1* также может быть каталогом; в таком случае необходимо задать параметр `-r`, чтобы можно было скопировать каталог и все его подкаталоги;
- *имя_файла2* не нужно указывать, когда имя файла должно остаться прежним; в таком случае файл будет скопирован в домашний каталог пользователя user2.

  Вместо *имя_файла2* также можно задать целевой каталог, применив, как обычно, символ ~ для обозначения домашнего каталога user2.

В завершение приведу еще один пример: предположим, что пользователь gabi работает на компьютере uranus. Он хочет перенести файл `abc.txt` в каталог `~/efg` на компьютере mars. Команда `scp` будет выглядеть так:

```
gabi@uranus$ scp abc.txt mars:~/efg/
gabi@mars's password: ******
```

SFTP (Secure FTP) — это основанный на SSH безопасный вариант протокола FTP. В следующем подразделе мы рассмотрим SFTP подробнее, коснувшись при этом вопросов передачи файлов через FTP и HTTP.

SSH-туннель

Опытные пользователи Linux могут применять SSH для создания туннелей. Конечно, машина или поезд по такому туннелю не пройдет, но зато по нему можно передавать любые IP-пакеты, направленные к определенному порту. Туннель SSH — это надежный путь, позволяющий переправлять IP-пакеты между компьютерами даже тогда, когда между этими компьютерами стоит брандмауэр, блокирующий порт.

Если туннель создается с клиентского компьютера, применяется параметр `-L localport:remotehost:remoteport`. Например, следующая команда позволяет получить доступ к порту 3306 компьютера mars через порт 3307 локального компьютера. Эта же команда одновременно запускает SSH-соединение, но вы можете воспрепятствовать этому с помощью параметра `-N` (если вам необходим только туннель, а оболочка не нужна). Если требуется войти в систему на компьютере mars под другим именем, следует указать имя для входа в систему как обычно: `-l name` или вот так: `name@remotehost`.

```
user@uranus$ ssh -L 3307:localhost:3306 username@mars
user@mars's password: ******
```

Туннель будет открыт до тех пор, пока вы не завершите SSH-соединение, нажав Ctrl+D. Если вы запустили `ssh` с параметром `-N`, выход из программы осуществляется нажатием Ctrl+C.

Как правило, MySQL используется через порт 3306. Теперь вы можете получить доступ с компьютера uranus к серверу MySQL, работающему на mars, через порт 3307 компьютера uranus. Вместе с командой `mysql` необходимо указать порт 3307 и имя хоста 127.0.0.1, чтобы SSH-туннель действительно использовался (по умолчанию MySQL устанавливает локальные соединения через сокет-файл).

```
user@uranus$ mysql -u mysqllogin -P 3307 -h 127.0.0.1 -p
Enter password: *******
```

Чтобы логин к MySQL работал, необходимо предварительно выполнить два условия. Сервер MySQL должен принять IP-соединения, установленные на компьютере mars. (Из соображений безопасности MySQL-сервер может иметь такую конфигурацию, при которой соединения могут устанавливаться только через сокет-файлы. В таком случае туннель не поможет, так как им можно соединять только порты.) В качестве хост-имени применяется имя компьютера, к которому ssh построила туннель — в данном случае mars (или mars.sol, если использовать домен sol, как и в других примерах этой книги).

SSH-туннели могут применяться для решения гораздо более сложных и разнообразных задач. Например, туннель можно использовать для построения *виртуальной частной сети*. Подробная документация приводится на сайте http://www.tldp.org/HOWTO/VPN-HOWTO/.

Файловая система SSH

С помощью команды sshfs, которая во многих дистрибутивах находится в одноименном пакете, можно интегрировать в локальное дерево каталогов файловую систему удаленного компьютера. Так можно, например, упростить процесс резервного копирования. Но учитывайте, что из-за шифрования всех данных файловая система SSH обычно характеризуется более низкой производительностью, чем, например, Samba или NFS. Поэтому возможности применения файловой системы SSH в локальных сетях весьма ограничены.

```
root# mkdir /media/ext-host
root# sshfs user@hostname /media/ext-host
root# ...
root# umount /media/ext-host
```

10.3. Передача файлов (FTP)

Основы FTP

FTP означает File Transfer Protocol (Протокол передачи файлов). Это достаточно старый способ передачи файлов по Сети. Своей популярностью FTP обязан методу Anonymous FTP (анонимный FTP-доступ): многие крупные серверы Интернета предоставляют своим пользователям доступ к так называемым FTP-архивам. Такой доступ не требует пароля (в отличие от обычного FTP).

Серьезный недостаток FTP заключается в том, что при входе в систему имя пользователя и пароль передаются в незашифрованном виде. Более надежная альтернатива — SFTP (Secure FTP), основанный на SSH. В качестве альтернативы для FTP также зачастую используется HTTP, то есть протокол для передачи сайтов.

В этой главе обсуждается только применение FTP, то есть проблема рассматривается с клиентской точки зрения. Чтобы FTP функционировал, на удаленной станции должен работать FTP-сервер. Его конфигурация описана в разделе 23.7.

Команда FTP

Предком всех FTP-клиентов является интерактивная текстовая команда `ftp`. Поскольку файлы обычно передаются из текущего каталога или в этот каталог, то перед запуском `ftp` нужно перейти в желаемый рабочий каталог командой `cd`. Чтобы начать FTP-сессию, вводится команда `ftp` *`имя_пользователя@ftp-сервер`* или просто `ftp` *`ftp-сервер`*. Если вы желаете работать с анонимным FTP, укажите в качестве имени пользователя слово `anonymous`.

Установив соединение и введя пароль, можно начинать работу: команды `cd`, `pwd` и `ls`, значения которых такие же, как в Linux, позволяют переходить между каталогами FTP-архива. Чтобы перенести файл из FTP-архива в текущий каталог вашего компьютера, выполните команду `get` *`файл`*. При этом имя файла не изменяется. И наоборот: с помощью `put` можно перенести файл из текущего каталога вашего компьютера в каталог FTP-архива. Разумеется, это работает лишь в том случае, когда у вас есть право на внесение изменений в этот каталог. При использовании анонимного FTP это обычно касается только каталогов с названием типа `/pub/incoming`. FTP-соединение завершается командой `quit` или `bye`. Важнейшие FTP-команды приводятся в табл. 10.1.

Таблица 10.1. Команды для работы с FTP

Команда	Функция
?	Вывод списка всех команд FTP
!	Выполнение команд оболочки
ascii	Переход в текстовый режим
binary	Переход в двоичный режим
bye	Завершение работы FTP
cd dir	Переход в указанный каталог FTP
close	Завершение соединения с сервером FTP
get file	Передача файла из архива FTP в текущий каталог
help «команда»	Вывод краткой справочной информации по указанной команде
lcd dir	Смена текущего каталога на локальном компьютере
ls	Вывод списка всех файлов, находящихся на FTP-сервере
lls	Вывод списка всех файлов, находящихся на локальном компьютере
mget *.pattern	Передача всех файлов, отвечающих образцу, из FTP-архива в текущий каталог
open	Соединение со сторонним компьютером (если первая попытка не удалась)
prompt	Активация/деактивация автоматического запроса о подтверждении передачи любого файла с помощью mget
put «файл»	Передача файла из текущего каталога в FTP-архив (upload)
quit	Завершение работы FTP

Команда	Функция
reget «файл»	Продолжение передачи файла, который ранее был передан частично
user	Возможность ввести новый логин

ПРИМЕЧАНИЕ

Прежде чем передать файл, необходимо перейти в двоичный режим с помощью команды binary. При работе в текстовом режиме FTP интерпретирует файлы как текст и пытается преобразовать файл в формат конкретного компьютера. После такого преобразования работать с двоичными файлами уже нельзя (к счастью, конфигурация большинства FTP-серверов имеет установку binary по умолчанию).

В следующих строках показано, как загрузить код ядра Linux с FTP-сервера:

```
user$ cd ~/src
user$ ftp ftp.kernel.org
Connected to zeus-pub.kernel.org.
220 Welcome to ftp.kernel.org.
Name (ftp.kernel.org:kofler): anonymous
331 Please specify the password.
Password: name@mysite.de
Using binary mode to transfer files.
ftp> cd pub/linux/kernel/v3.0
250 Directory successfully changed.
ftp> ls
...
ftp> get linux-3.5.tar.bz2
local: linux-3.5.tar.bz2 remote: linux-3.5.tar.bz2
227 Entering Passive Mode (204,152,191,5,20,69)
150 Opening BINARY mode data connection for linux-3.5.tar.bz2 (69305709 bytes).
...
ftp> quit
```

Другие программы для работы с FTP

Работать с командой `ftp` достаточно неудобно. К счастью, альтернативных программ достаточно много.

- Браузеры, файловые менеджеры — все браузеры и файловые менеджеры, используемые в Linux, могут применяться и для закачки FTP. Некоторые программы даже обеспечивают удобную закачку (например, Nautilus и Konqueror).
- Графические FTP-клиенты — такие программы, как `gftp` (Gnome), оптимизированы под выполнение типичных FTP-задач. В них есть специальные функции, например для управления закладками и паролями, обеспечивающие передачу нескольких файлов, синхронизацию каталогов и т. д.
- `ncftp` — хотя эта альтернативная программа имеет текстовый пользовательский интерфейс, работать с ней удобнее, чем с `ftp`.
- `sftp` — данная программа так же минималистична, как и `ftp`, но при этом она заметно надежнее. При ее использовании на удаленной станции обязательно должен работать SSH-сервер (а не FTP-сервер).

- `wget`, `curl`, `lftp`, `rsync`, `mirror` и `sitecopy` — эти команды полезны при автоматической передаче файлов или целых деревьев каталогов через FTP.

FTP-адрес с паролем

Если вы не хотите работать с протоколом FTP под именем `anonymous`, а желаете войти в систему под собственным именем и паролем, то в большинстве клиентов FTP будет использоваться следующий синтаксис:

`ftp://`*`имя_пользователя:пароль@имя_сервера`*

Пассивный режим. Некоторые FTP-клиенты могут работать неправильно, если между вашим компьютером и FTP-сервером находится брандмауэр или если вы работаете в локальной сети, которая соединена с Интернетом с использованием маскарадинга. В таких случаях почти всегда помогает перевод клиента в так называемый пассивный режим. К сожалению, единой команды для этого не существует, поэтому придется изучить документацию (большинство клиентов самостоятельно опознают такую ситуацию и автоматически переходят в пассивный режим).

SFTP

Команда `sftp` входит в состав пакета `openssh`. Внутри программа `sftp` использует принципиально иной протокол, нежели `ftp`, и может применяться в качестве `ssh` только при условии, что на удаленной станции работает SSH-сервер. Анонимный FTP с `sftp` применять нельзя. За исключением этого, работа с программой практически не отличается от работы с `ftp`. С помощью команды `sftp -b` *`batch-файл`* можно автоматизировать процесс SFTP-закачек.

Альтернативы для SFTP. Многие считают `sftp` слишком «спартанским» вариантом. В любом случае выбор SFTP-клиентов значительно уже, чем при работе с FTP. Кроме того, иногда требуется немного подождать, чтобы соединение установилось.

- `gftp` — предлагает многочисленные возможности конфигурирования SFTP (FTP ▸ Options ▸ SSH (FTP ▸ Настройки ▸ SSH)). Если возникнут сложности, проверьте, тот ли порт вы используете (для SSH — 22, а не 21, как для FTP). Часто может потребоваться установить флажок Use SSH2 SFTP Functions (Использовать функции SSH2 SFTP).
- KDE — с помощью Konqueror или Nautilus можно инициировать SFTP-соединение, введя в адресную строку `sftp://`*`пользователь@имя_сервера`*. После запроса пароля в Konqueror можно работать как с обычными каталогами. Оба файловых менеджера также непосредственно поддерживают SSH-протокол, который функционирует даже при отсутствии `sftp`. В таком случае адрес вводится в таком формате: `fish://`*`пользователь@имя_сервера`*.

WGET

Интерактивное применение команды не подходит для автоматизации загрузок, допустим, в виде одного сценария. Кроме того, `ftp` не может похвастаться гибкостью.

Например, невозможно самостоятельно возобновить прерванную закачку. В данном случае будет полезна команда wget, специально разработанная для выполнения крупных загрузок или передачи целых деревьев каталогов. Эта команда в равной мере поддерживает протоколы FTP, HTTP и HTTPS.

Примеры. Программа wget скачивается на компьютер как самый обычный файл:

```
user$ wget ftp://myftpserver.de/имя.abc
```

Если закачка по каким-то причинам прерывается, ее можно без проблем возобновить с помощью параметра -c:

```
user$ wget -c ftp://myftpserver.de/имя.abc
```

На закачку очень больших файлов, например образов диска с дистрибутивами Linux, даже при хорошем соединении с Интернетом может уйти несколько часов, поэтому рекомендую ставить такие крупные загрузки на ночь. Следующая команда позволяет практически наверняка гарантировать, что наутро вы на самом деле найдете на компьютере файл, поставленный на закачку вечером. Благодаря параметру -t 20 закачка будет 20 раз возобновляться, если вдруг прервется, то есть файл успешно закачается и после 20 обрывов связи. Благодаря --retry-connrefused даже при возникновении ошибки connection refused (в соединении отказано) будет предпринята новая попытка соединения. Это полезно в тех случаях, когда сервер загрузок наверняка ненадежен и определенно будет ненадолго недоступен время от времени.

```
user$ wget -t 20 --retry-connrefused http://mydownloadserver.de/имя.iso
```

Следующая команда скачивает все файлы, необходимые для того, чтобы можно было прочитать указанный сайт в неизменном виде, будучи отключенным от Интернета. Коротко о параметрах: -p скачивает все файлы таблиц стилей и изображения; -k преобразует ссылки, содержащиеся в скачанных файлов, так, чтобы они указывали на соответствующие файлы локального компьютера; -E добавляет к скачанным файлам сценариев (ASP, PHP) расширение HTML; -H обеспечивает переход по ссылкам на внешние сайты.

```
user$ wget -p -k -E -H http://mysite.com/site.html
```

Если вы хотите прочитать в офлайн-режиме целый сайт, воспользуйтесь следующей рекурсивной командой для скачивания (параметр -r). С помощью параметра -l 4 мы ограничиваем глубину рекурсии четырьмя уровнями.

```
user$ wget -r -l 4 -p -E -k http://mysite.com
```

Команда curl

Команда curl помогает при передаче файлов с сервера или на сервер FTP, HTTP или других типов. На соответствующей странице справки man приводится внушительный список протоколов, с которыми может работать curl. В этом разделе мы рассмотрим только закачки через FTP. При программировании сценариев особенно практичной является функция curl, помогающая обрабатывать данные стандартного ввода либо записывать информацию в стандартный вывод. Иначе говоря,

вам не потребуется предварительно создавать файл *.tar.gz, а потом передавать его на FTP-сервер; обе операции можно будет выполнить одновременно, разделив в коде соответствующие записи вертикальной чертой.

Следующая команда передает указанный файл на FTP-сервер backupserver и сохраняет этот файл в каталоге dir:

```
user$ curl -T файл -u имя_пользователя:пароль ftp://backupserver/dir
```

Для обработки данных из стандартного канала ввода добавьте к -T в имени файла дефис. Следующая команда сохраняет результат, получаемый после выполнения команды tar, прямо в файл name.tgz на FTP-сервере:

```
user$ tar czf - dir/ | curl -T - -u usern:pw ftp://bserver/имя.tgz
```

Программа lftp

Программа lftp является удачным интерактивным FTP-клиентом. Она также хорошо подходит и для того, чтобы с удобством выполнять FTP-закачки и другие команды в рамках одного сценария. При работе с lftp можно использовать параметр -c, вместе с которым вы передаете несколько FTP-команд, разделенных точками с запятой, либо параметр -f, после которого указывается файл, где построчно перечислены необходимые команды. При этом первой всегда будет идти команда вида user *имя_пользователя*,*пароль имя_сервера*, устанавливающая соединение с FTP-сервером. Следующая команда демонстрирует закачку файла:

```
root# lftp -c "open -u имя_пользователя,пароль backupserver; put www.tgz"
```

Если вы хотите дать новое имя файлу, находящемуся на FTP-сервере, дополнительно укажите параметр -o *новое_имя*. В ходе закачки lftp будет демонстрировать ход выполнения задачи.

Для переноса на резервный сервер целого каталога воспользуйтесь командой mirror -R. Как правило, команда mirror копирует каталоги с FTP-сервера на локальный компьютер. Параметр -R позволяет запустить обратный процесс. Вот пример:

```
root# lftp -c "open -u usern,passw bserver; mirror -R directory"
```

В отличие от других FTP-клиентов, lftp поддерживает команду du, с помощью которой можно определить, сколько места на диске уже занимают ваши резервные файлы. Это может быть важно, если дисковое пространство на сервере для резервного копирования строго ограничено. Следующая команда позволяет установить, сколько дискового пространства имеется в наличии, не прибегая к интерактивному вмешательству. Параметр -s указывает, что вас интересует только конечный результат. Параметр -m говорит о том, что в качестве единицы измерения будет применяться мегабайт.

```
user$ lftp -c "open -u имя_пользователя,пароль bserver; du -s -m"
2378  .
```

Если вы хотите использовать результат в дальнейших вычислениях, второй столбец (в котором находится точка, указывающая, что числовое значение отно-

сится к текущему каталогу) вам помешает. Добавьте к команде `cut -f 1`, чтобы извлечь первый столбец:

```
user$ lftp -c "open -u usern,passw bserver; du -s -m" | cut -f 1
2378
```

Команды rsync, mirror, sitecopy

Команда `rsync` помогает копировать или синхронизировать целые каталоги. Подробное описание этой команды приводится в разделе 25.4. Если на парном компьютере не работает ни SSH-, ни `rsync`-сервер, вместо `rsync` можно воспользоваться командами `mirror` или `sitecopy`.Сценарий `mirror`, написанный на языке Perl из одноименного пакета, копирует целые деревья каталогов с FTP-сервера на локальный компьютер. Команда `sitecopy`, напротив, оптимизирована именно для того, чтобы закачать каталог на веб-сервер. При этом передача информации может осуществляться по вашему выбору через FTP или WebDAV,

11 Базовая конфигурация

Теперь обратимся к конфигурированию системы Linux — этой важной теме будут посвящены несколько следующих глав. Здесь я приведу некоторую вводную информацию и затрону такие темы:

- конфигурирование текстовых консолей;
- настройка даты и времени;
- управление пользователями;
- интернационализация, кодировка, Unicode;
- обзор конфигурации аппаратного обеспечения;
- файлы регистрации (журналирование).

В следующих главах будет рассмотрено управление пакетами, системными библиотеками, графическими системами (X), администрирование файловой системы, запуск системы (GRUB, Init-V, Upstart), работа с ядром и его модулями.

11.1. Введение

В этой и следующей главах мы заглянем «за кулисы» системы и увидим, как функционирует программа конфигурации Linux. Нам предстоит разобраться, что, где и как работает, как осуществляется управление системой и как задаются настройки по умолчанию. Кроме того, в этих главах я предоставлю вам много полезных фоновых сведений о работе системы в целом.

К сожалению, каждый из множества дистрибутивов конфигурируется по-своему, не так, как остальные. И различия обычно заключаются в мелочах. В этой книге я попытался привести все к общему знаменателю и описать как можно больше систем. И все же не исключено, что именно в вашем дистрибутиве некоторые детали будут построены чуть по-другому. В таких случаях рекомендую обратиться к документации.

Вы можете не согласиться со мной, но я убежден, что при описании администрирования именно комплексный подход является оптимальным, то есть не стоит писать одну книгу об администрировании Red Hat, другую — о SUSE и т. д. В первую очередь потому, что даже отдельные дистрибутивы изменяются от версии к версии. Рано или поздно вам обязательно придется учиться, по крайней мере читать и понимать англоязычные руководства, файлы справки и т. д. Эта книга не

заменит оригинальных пособий или пошаговых руководств, однако поможет вам приобрести необходимые фоновые знания.

Кто здесь системный администратор?

Итак, до сих пор системный администратор был для вас таинственным незнакомцем, который снова и снова приходил к вам на помощь, — часто нехотя, иногда очень усталый. Если вы не работаете на крупном предприятии, то администратор вообще становится для вас кем-то абстрактным, героем книг, в которых написано: «Если что-то не ладится, обратитесь к системному администратору».

Если вы сами установили Linux на своем компьютере, то картина меняется: теперь вы сами себе системный администратор! Не пугайтесь такого громкого титула: это просто специалист, который сам занимается конфигурацией своего компьютера. И если спектр задач ограничивается основными функциями Linux, то работа будет под силу каждому. Правда, вам постоянно придется заглядывать в книги по Linux, например в эту или другие специальные источники.

Конфигурационные инструменты и программы для администрирования

В большинстве дистрибутивов имеются очень удобные конфигурационные программы, с которыми можно работать как в ходе установки, так и после нее: `drakconf` в Mandriva, различные программы `system-`*`xxx`* в Red Hat или Fedora, модули YaST в SUSE и т. д. При конфигурировании системы к этим инструментам необходимо прибегать в первую очередь! Они оптимизированы под конкретный дистрибутив и избавляют вас от лишней работы.

Наряду с многочисленными конфигурационными программами, поставляемыми вместе с дистрибутивами, есть и внешние инструменты. Кроме того, в некоторых дистрибутивах Linux имеются собственные инструменты для администрирования. В табл. 11.1 приведен обзор наиболее популярных подобных инструментов. Тем не менее, важно учитывать, что область применения и функционал отдельных инструментов из этого списка сильно различаются. Часть из перечисленных инструментов специально предназначена для поддержки большого количества однородных инсталляций Linux в сети. Звездочка в столбце «Бесплатный» означает, что данная программа распространяется свободно и ею можно бесплатно пользоваться не только на домашнем компьютере, но и на предприятии. (Для большинства коммерческих программ существуют и бесплатные варианты, которые отличаются от платных лишь меньшим набором функций.)

Таблица 11.1. Некоторые инструменты для администрирования

Ссылка	Описание функции	Бесплатный
http://webmin.com/	Администрирование системы и сети	*
http://fai-project.org/	Установка и распределение программного обеспечения	*

Продолжение ⇩

Таблица 11.1 (продолжение)

Ссылка	Описание функции	Бесплатный
http://m23.sourceforge.net/	Распределение и администрирование программного обеспечения	
http://directory.fedoraproject.org/	Пользовательский интерфейс для RHEL/Fedora, работающий по протоколу LDAP	*
https://oss.gonicus.de/labs/gosa/	Создание учетной записи для управления Debian по LDAP	*
http://redhat.com/rhn/	Администрирование Red Hat	
http://novell.com/zenworks/	Администрирование Novell/SUSE	
http://canonical.com/landscape/	Администрирование Ubuntu	
http://univention.de/produkte/ucs/	Конфигурирование сервера локальной сети и почтового сервера	
http://zentyal.org/	Конфигурирование сервера локальной сети (ранее — eBox)	
http://cpanel.net/	Администрирование корневого и веб-сервера	
http://parallels.com/plesk/	Администрирование корневого и веб-сервера	

СОВЕТ

При применении коммерческих инструментов для администрирования легко завязнуть в новых зависимостях. Опыт показывает, что очень многие такие инструменты приходят и уходят, как будто их и не было. Поэтому с осторожностью пользуйтесь внешними инструментами для администрирования!

Конфигурационные файлы

Продуманные до мелочей конфигурационные инструменты с красивыми пользовательскими интерфейсами разработаны именно для того, чтобы облегчить вам работу при непосредственном изменении конфигурационных файлов Linux. Это исключительно удобно как раз для новичков Linux.

Однако есть множество причин, по которым я рекомендую вам вплотную заняться конфигурационными файлами, — а значит, познакомиться с внутренней архитектурой Linux.

- Изменять конфигурационные файлы можно в любом текстовом редакторе, в том числе и тогда, когда вы работаете с текстовой консолью или по сети через SSH. Когда вы разберетесь, как конфигурируется та или иная функция Linux, вы сможете применить эти знания при работе с практически любым другим дистрибутивом Linux. А работа со многими конфигурационными инструментами зависит от дистрибутива.
- Только внося изменения непосредственно в конфигурационные файлы, вы сможете управлять всеми аспектами системных функций. А функциональность конфигурационных инструментов часто ограничена лишь немногими (самыми важными) явлениями.
- Конфигурационные файлы можно с легкостью копировать с одного компьютера на другой. Это помогает сэкономить массу времени, когда вы переходите

к работе с новым дистрибутивом, заново устанавливаете Linux на другой компьютер и т. д.

- Чем лучше вы понимаете строение конфигурационных файлов и связанные с ними возможности управления, тем больше вы понимаете саму Linux и тем реже компьютер оказывается «черным ящиком», в который невозможно заглянуть.

Практически все конфигурационные файлы Linux находятся в каталоге `/etc`.

Связанные друг с другом конфигурационные файлы крупных программ часто располагаются в отдельных подкаталогах. Например, все конфигурационные файлы графической системы X размещены в каталоге `/etc/X11`. Некоторые подкаталоги имеют особое значение:

- `/etc/default` — файлы, характерные для отдельных дистрибутивов (Debian, Ubuntu);
- `/etc/init.d` или `/etc/rc.d` — система Init-V (запуск системы, см. главу 16);
- `/etc/init` — Upstart (см. главу 16);
- `/etc/init` — Systemd (см. главу 16);
- `/etc/sysconfig` — файлы, специфичные для отдельных дистрибутивов (Fedora, Red Hat, SUSE).

Рекомендуется создавать резервную копию всего каталога `/etc`. Тогда при внесении изменений вы в любой момент сможете оперативно узнать, как выглядел конфигурационный файл в исходном состоянии.

```
root# mkdir /etc-backup
root# cp -a /etc/* /etc-backup
```

СОВЕТ

При редактировании конфигурационных файлов следите за тем, чтобы последняя строка завершалась нажатием клавиши Enter. Если такое окончание отсутствует, некоторые программы Linux могут допускать ошибки при обработке файлов.

Поиск конфигурационных файлов. Если вы не можете найти в своем дистрибутиве какой-либо конфигурационный файл, на это может быть много причин: возможно, у вас не установлены базовые программные пакеты или же конфигурационные файлы расположены в вашем дистрибутиве где-то в другом месте. При поиске пользуйтесь командами `locate`, `find` и `grep`. Следующая команда показывает, как выполнить поиск файлов, в названиях которых содержится последовательность символов `abcde`, в `/etc` и во всех его подкаталогах:

```
root# cd /etc
root# find -type f -exec grep -q abcde {} \; -print
```

Работая над книгой, мне часто приходилось искать, как в дистрибутиве производится управление функцией *y* или как вызвать функцию *z*. Для этого вышеуказанную команду пришлось ввести не одну сотню раз. Чтобы сэкономить время и силы, я написал маленький сценарий `grepall`, выполняющий эту работу.

Как активизировать новую конфигурацию. В некоторых программах изменения, внесенные в конфигурацию, вступают в силу лишь после повторного запуска

программы либо после того, как вы прямо скомандуете системе заново считать конфигурационные файлы. Обычно для этого требуется выполнить одну из двух следующих команд:

```
root# /etc/init.d/funktionsname restart  (Debian 6, все дистрибутивы)
root# /etc/init.d/funktionsname reload
root# service funktionsname restart      (Большинство распространенных дистрибутивов)
root# service funktionsname reload
```

В отличие от Windows, в Linux практически никогда не требуется перезапускать компьютер. Исключение — необходимость внесения изменений в ядро, а также некоторые аппаратно-зависимые настройки, которые можно произвести лишь непосредственно при запуске системы.

11.2. Конфигурация текстовых консолей

В современных дистрибутивах Linux сразу запускается графическая система, и новички порой даже не знают, что в системе есть текстовая консоль. Разумеется, сплошь и рядом встречаются ошибки в конфигурации X либо работа с графической системой может быть не предусмотрена по каким-либо другим причинам. При некоторых вариантах установки сервера приходится даже намеренно отказываться от работы с графической системой. Именно в таких случаях вам пригодится умение пользоваться консолью.

За простейшие настройки, такие как раскладка клавиатуры или вид шрифта, отвечает система `kbd` или более новый вариант `console` — в зависимости от дистрибутива. Однако в каждом дистрибутиве отдельные детали конфигурации различаются.

Раскладка клавиатуры

Debian, Ubuntu. в Debian и Ubuntu за раскладку клавиатуры отвечают программы из пакета `console-setup`. Конфигурационный файл `/etc/default/console-setup` управляет гарнитурой шрифта, файл `/etc/default/keyboard` — раскладкой клавиатуры. Здесь при конфигурации применяются те же параметры, что и в системе X:

```
# /etc/default/console-setup
...
# Гарнитура шрифта
CHARMAP="UTF-8"
CODESET="Lat15"
FONTFACE="VGA"
FONTSIZE="16"
# /etc/default/keyboard
# Клавиатура
XKBMODEL="pc105"
XKBLAYOUT="de"
XKBVARIANT=""
XKBOPTIONS="lv3:ralt_switch"
```

В Debian за интерпретацию обоих файлов отвечает сценарий /bin/setupcon, который в свою очередь выполняется сценарием Init-V /etc/init.d/keyboard-setup.

В Ubuntu интерпретация конфигурационного файла console-setup обеспечивается системой udev (если быть точным — сценарием /lib/udev/console-setup-tty). Конфигурацию клавиатуры выполняет Upstart (/etc/init/console-setup.conf). При этом считывается не /etc/ default/keyboard, а файл /etc/console-setup/cached.kmap.gz. Данный файл обновляется на этапе создания файла Initrd, это делается сценарием /usr/bin/ckbcomp.

И в Debian, и в Ubuntu действует рекомендация: изменять раскладку клавиатуры следует по возможности не путем непосредственного изменения конфигурационных файлов, а с помощью следующей команды:

```
root# dpkg-reconfigure keyboard-configuration
```

Так вы гарантируете, что все файлы будут обновлены и все настройки сразу же вступят в силу.

Fedora. В Fedora для настройки раскладки клавиатуры используется пакет kbd. Таблица соответствий устанавливается во время запуска системы сценарием Initrd. Раскладка клавиатуры определяется в конфигурационном файле /etc/sysconfig/keyboard:

```
# Файл /etc/sysconfig/keyboard
KEYTABLE="de-latin1-nodeadkeys"
MODEL="pc105"
LAYOUT="de"
VARIANT="nodeadkeys"
```

Изменения, внесенные в этот файл, вступают в силу лишь тогда, когда файлы Initrd создаются заново, это делается с помощью команды dracut.

```
root# dracut -f
```

Кроме того, файл /etc/sysconfig/keyboard также интерпретируется программой fedora-setup-keyboard для конфигурации клавиатуры под X. В системе X эта программа выполняется на уровне запуска 5 и слушает /etc/sysconfig/keyboard. При каждом изменении заново создается файл /etc/X11/ xorg.conf.d/00-system-setup-keyboard.conf.

SUSE. В SUSE, как и в Fedora, используется пакет kbd и конфигурационный файл /etc/sysconfig/keyboard. На этом сходство заканчивается. Конфигурационный файл интерпретируется сценарием Init-V /etc/init.d/kbd. Настройки действуют только для консоли, а не для X.

Гарнитура шрифта

Как правило, консоли совместимы с Unicode. Но в любом случае максимально возможное количество символов в шрифтах консолей очень невелико (256 или 512), поэтому шрифты консолей включают лишь малую толику символов Unicode.

Debian, Ubuntu. Настройки конфигурации находятся в файле /etc/default/console-setup и при запуске системы интерпретируются сценарием setupcon. В Debian

за эту операцию отвечает сценарий Init-V /etc/init.d/console-setup, в Ubuntu — конфигурационный файл Upstart /etc/init/console-setup.conf.

Fedora. В Fedora шрифт настраивается с помощью команды setfont сценарием файла Initrd, при этом интерпретируется конфигурационный файл /etc/sysconfig/i18n. Изменения, вносимые в этот файл, вступают в силу только при создании новых файлов Initrd

SUSE. В SUSE шрифт консоли настраивается с помощью /etc/init.d/kbd. Этот сценарий интерпретирует /etc/sysconfig/console и настраивает шрифт командой setfont. По умолчанию используется шрифт lat9w-16.psfu, содержащий наряду с набором символов «латиницы-1» еще и символ евро.

Gpm-конфигурация (мышь)

Мышь может применяться только в системе X Window System. Но программа gpm позволяет в ограниченном объеме использовать мышь и в текстовых консолях: в частности, можно выделять текст левой кнопкой мыши и вставлять его правой или средней кнопкой в том месте, где стоит курсор. Однако обратите внимание, что в большинстве программ для консоли невозможно изменять положение курсора мышью.

Если программа gpm установлена в системе, то, как правило, она запускается системой Init. Конфигурация, в зависимости от дистрибутива, осуществляется в /etc/gpm.conf или /etc/sysconfig/mouse, причем как правило не требуется менять настройки, заданные по умолчанию.

11.3. Дата и время

Поскольку компьютерные сети раскинулись по всему миру, необходимо пользоваться единым международным временем. Таким временем является GMT (Greenwich Mean Time, среднее время по Гринвичскому меридиану). На всех компьютерах UNIX это время является «мерой всех вещей». Еще это время обозначается сокращением UTC (Universal Time, Coordinated, универсальное координированное время).

При сохранении файла компьютер не фиксирует текущее местное время, а пересчитывает время по этому международному стандарту. Если теперь просмотреть файл с помощью команды ls -l, то время будет пересчитано как местное для того региона, где находится компьютер. Этот метод позволяет определить, какой файл актуальнее: тот, что был сохранен в Мюнхене в 18:00 по местному времени, или тот, что сохранили в 12:30 в Нью-Йорке, — тоже по местному времени.

Настройка времени при запуске компьютера. На первом этапе работы процесса Init-V настраивается компьютерное время. Для этого команда hwclock считывает время с часов CMOS компьютера. Часы CMOS могут содержать как местное время, так и GMT. Если компьютер работает и с Windows, удобнее настроить часы CMOS на местное время.

В Red Hat, Fedora и SUSE в конфигурационном файле /etc/sysconfig/clock содержится информация о том, на какое время настроены часы CMOS — на местное

или GMT — и в каком часовом поясе находится компьютер. В Debian и Ubuntu в файле /etc/default/rcS сообщается, на какое время настроены часы CMOS, а информация о часовом поясе содержится в файле /etc/timezone.

Настройка часового пояса. Чтобы команды вроде ls или файловый менеджер KDE либо Gnome пересчитывали GMT в местное время и отображали его в таком виде, каждая программа должна знать, в каком часовом поясе она работает.

Для этого почти во всех программах Linux используются функции библиотеки glibc. Она интерпретирует файл /etc/localtime, являющийся копией файла часового пояса (time zone file) из каталога /usr/share/zoneinfo. Файл localtime также может быть символьной ссылкой на файл часового пояса. Чтобы заново настроить временной пояс, просто скопируйте нужный файл часового пояса в /etc/localtime:

```
root# cp /usr/share/zoneinfo/Europe/Berlin /etc/localtime
```

В Debian и Ubuntu часовой пояс также задается в текстовом файле /etc/timezone. Но этот файл интерпретируется не непосредственно библиотекой glibc, а лишь инструментами, предназначенными для новой настройки файла localtime.

Конфигурационные инструменты. В зависимости от дистрибутива для настройки часового пояса или для изменения времени и даты на часах компьютера можно применять различные конфигурационные программы:

- Gnome 2.*n* — time-admin;
- Gnome 3.*n* — модуль системных настроек Дата и время;
- KDE — модуль центра управления Управление системой ▸ Дата/Время;
- Debian, Ubuntu — tzconfig, инструменты Gnome/KDE;
- Red Hat/Fedora — system-config-date;
- SUSE — YaST-модуль Система ▸ Часовой пояс.

NTP (Сетевой протокол времени)

Чтобы не работать с CMOS-часами компьютера, которые печально известны своей неточностью, можно сверять время с сервером времени в Интернете. Это делается по протоколу сетевого времени (Network Time Protocol, NTP). Существует два способа использования NTP.

- Команда ntpdate *однократно* сверяется с точным временем и настраивает часы компьютера. В случае с компьютерами, которые часто приходится включать и выключать, это обеспечивает приемлемую точность часов.
- На сервере, который может без перерывов работать в течение целых недель или месяцев, недостаточно один раз правильно установить часы. Время на компьютере постепенно будет все сильнее отличаться от точного времени. В данном случае поможет демон ntpd, регулярно устанавливающий контакт с другими серверами времени и каждый раз немного корректирующий локальное время компьютера. Одновременно ntpd может сам играть роль сервера времени для других компьютеров (например, для всех клиентов, работающих в локальной сети).

Даже если вы используете на компьютере ntpd, команда ntpdate удобна, чтобы точно установить время при первом запуске. Ntpd же функционирует именно тогда, когда изначальное расхождение между точным и локальным временем не превышает минуты.

Вместо классических инструментов для работы с NTP (ntpdate, ntpd) для установки времени также можно использовать более современную программу Chrony (подробнее о ней — в следующем подразделе). Chrony также основана на NTP и по умолчанию применяется в Fedora начиная с 16 версии и выше.

ВНИМАНИЕ

Существует большое количество программ, которые плохо воспринимают резкие изменения настроек времени. К ним относятся, например, аутентификационная система Kerberos, POP3- и IMAP-серверы Dovecot, а также серверы баз данных. Перед выполнением ntpdate или автоматической настройкой времени рекомендуется завершать такие программы и после запускать заново. Dovecot автоматически завершается, если обнаруживает, что часы переведены назад.

Ссылки. Подробнее об управлении датой и временем в Linux рассказано на следующих сайтах: http://www.ntp.org/; http://tldp.org/HOWTO/TimePrecision-HOWTO/.

Debian, Ubuntu. То, как запускать программы ntpdate и ntpd и запускать ли их вообще, зависит от конкретного дистрибутива. В Debian и Ubuntu ntpdate устанавливается по умолчанию и всякий раз выполняется в тех случаях, когда устанавливается сетевое соединение (сценарий /etc/network/if-up.d/ntpdate).

Если вы также хотите выполнить на компьютере ntpd, нужно установить пакет ntp. Кроме того, в файле /etc/ntp.conf потребуется указать адрес близлежащего, хорошо доступного NTP-сервера.

С помощью ntpq -p можно удостовериться, что ntpd работает. В выводе этой команды самая важная информация содержится в столбце offset. Здесь указывается разница между локальным временем и временем на различных справочных серверах (разница дается в миллисекундах). Это значение должно быть как можно меньше. Чтобы ntpq -p выдавала результаты, имеющие практическую ценность, программа ntpd должна какое-то время поработать (хотя бы несколько минут). Обратите внимание на то, что при небольших отклонениях ntpd не просто корректирует показатель времени, а заставляет часы идти немного медленнее или немного быстрее, пока не будет достигнуто верное значение. Таким образом, удается избегать резких изменений во времени.

```
root# ntpq -p
   remote   refid st t  when  poll  reach delay offset jitter
==============================================================================
europium.canoni ...      2  u  2     64    1     21.565 -117.64 0.002
www.alter-provi ...      2  u  1     64    1     20.436 -118.56 0.002
```

Если отклонение во времени превышает одну секунду (то есть в столбце offset будет значение более 1000), потребуется установить время вручную с помощью ntpdate:

```
root# service ntp stop
root# ntpdate de.pool.ntp.org
root# service ntp start
```

Red Hat

В RHEL 6 конфигурация NTP осуществляется с помощью `system-config-date`. Когда вы активируете в этой программе NTP, при запуске компьютера также запускается `ntpd`. С помощью `ntpq -p` вы можете убедиться, что все функционирует правильно. Если начальное отклонение от точного времени окажется слишком большим, то нужно ненадолго остановить `ntpd` и синхронизировать локальное время с помощью `ntpdate`.

SUSE

В SUSE конфигурация NTP проходит в два этапа. Сначала в YaST-модуле SYSTEM ▸ Date and Time (Система ▸ Дата и время) можно задать точное время с помощью NTP (кнопка Change (Изменить)). При этом один раз выполняется `ntpdate`. Чтобы настроить NTP-сервер, запустите YaST-модуль Network Services ▸ NTP Setting (Сетевые службы ▸ Настройка NTP) и установите флажок Start NTP now and by booting (Запускать NTP сейчас и при загрузке). Очень важно, сколько серверов времени вы указываете при настройке — один или несколько. (По умолчанию применяются только локальные часы, а этого недостаточно.) `Ntpd` будет после этого запускаться сценарием Init-V `ntp`.

Chrony

В версиях Fedora 16 и выше разработчики заменили классический NTP-демон новой программой Chrony. Она особенно хороша при работе с ноутбуками и виртуальными машинами, которые связаны с Интернетом не постоянно и время на которых часто приходится точно корректировать после долгих периодов работы в офлайне. Конфигурация осуществляется в `/etc/chrony.conf`. Если автоматическая установка времени после долгих периодов работы в офлайне будет функционировать неправильно, выполните следующую команду:

```
root# service chronyd restart
```

Более подробная информация по программе Chrony приводится на следующем сайте: http://chrony.tuxfamily.org/.

11.4. Пользователи и группы, пароли

Когда мы говорим об управлении пользователями, то подразумеваем в первую очередь управление тем, кто имеет доступ к каким файлам, какие программы имеет право выполнять, с какими файлами устройств работать и т. д. Управление пользователями и доступом требуется в тех случаях, когда на одном и том же компьютере могут работать несколько человек. Должны существовать правила, в соответствии с которыми один пользователь будет получать права читать и изменять файлы других пользователей.

В Linux для такого управления создается список пользователей. Кроме того, каждый пользователь может относиться минимум к одной, но, возможно, и сразу к нескольким группам. Группы позволяют нескольким пользователям получать доступ к общим файлам или программам.

Чтобы можно было наладить управление правами доступа, вместе с каждым файлом сохраняется и информация о его владельце, принадлежности владельца к группам и *биты доступа*. Поскольку программы — это тоже файлы, а доступ к аппаратным компонентам часто осуществляется через так называемые *файлы-устройства*, этот механизм практически универсален.

Конфигурационные программы

В принципе как администратор вы можете в значительной мере управлять пользователями вручную, непосредственно изменяя файлы, рассмотренные в этом разделе. Однако гораздо удобнее и надежнее освоить инструменты для управления группами и пользователями, входящие в состав большинства дистрибутивов:

- Gnome 2.*n* — `users-admin` (входит в состав `gnome-system-tools`);
- Gnome 3.*n* — модуль настройки системы Управление учетными записями;
- KDE — модуль центра управления Управление системой ▸ Управление пользователями;
- Debian, Ubuntu — инструменты Gnome или KDE;
- Red Hat, Fedora — `system-config-users` (рис. 11.1);
- SUSE — модуль YaST Безопасность ▸ Пользователи и безопасность ▸ Группы.

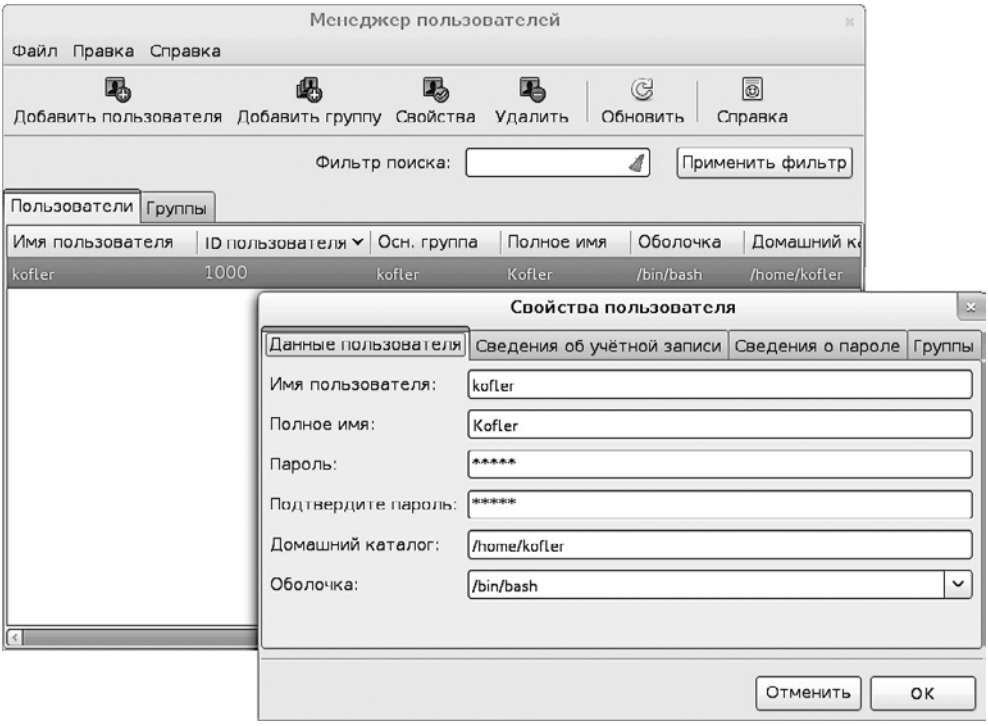


Рис. 11.1. Управление пользователями в Fedora

Команды

Если вы готовы отказаться от удобных пользовательских интерфейсов или хотите автоматизировать управление пользователями с помощью сценариев, можете воспользоваться командами, приведенными в табл. 11.2 в следующем примере показано, как создать учетную запись нового пользователя testuser и присвоить ему пароль.

```
root# useradd -m testuser
root# passwd testuser
New passwd: xxx
Re-enter new passwd: xxx
```

Таблица 11.2. Команды для управления пользователями и группами

Команда	Функция
adduser	Создает учетную запись нового пользователя (Debian)
addgroup	Создает новую группу (Debian)
chgrp	Изменяет групповую отнесенность файла
chmod	Изменяет биты доступа определенного файла
chown	Передает файл другому владельцу
chsh	Изменяет стандартную оболочку, применяемую при работе данным пользователем
delgroup	Удаляет группу (Debian)
deluser	Удаляет учетную запись пользователя (Debian)
groupadd	Создает новую группу
groupdel	Удаляет группу
groupmod	Изменяет свойства группы
groups	Показывает группы, к которым принадлежит данный пользователь
id	Показывает ID-номера текущей группы и текущего пользователя
newgrp	Изменяет активную группу определенного пользователя
newusers	Создает несколько новых пользовательских учетных записей
passwd	Изменяет пароль пользователя
useradd	Создает новую учетную запись пользователя
userdel	Удаляет учетную запись пользователя
usermod	Изменяет свойства учетной записи пользователя

Как вы, конечно же, заметили, для решения некоторых задач предусмотрены сразу по две команды (например, `adduser` и `useradd`). Команды `adduser`, `addgroup`, `deluser` и `delgroup` — это специфичные для Debian дополнения к общепринятым командам `useradd`, `groupadd` и т. д. в Debian, Ubuntu и других дистрибутивах, произошедших от Debian, эти команды учитывают правила, заданные в файлах `/etc/adduser.conf` и `/etc/deluser.conf`.

Как обычно, Red Hat и Fedora вносят в общую картину некоторую путаницу: в этих дистрибутивах также имеются команды `adduser`, `addgroup`, `deluser` и `delgroup`. Однако это совсем не те команды, которые используются в Debian, а всего лишь ссылки на `useradd`, `groupadd`, `userdel` и `groupdel`. Именно поэтому синтаксис команд `adduser` и `useradd` в Fedora идентичен — и при этом отличается от синтаксиса `adduser` в Debian!

SUSE/Novell также не ищут легких путей: там нет команд `adduser/-group` и `deluser/-group`. Для выполнения функций `groupadd/-del/-mod` и `useradd/-del/-mod` применяется собственная разработка, которая совместима с командами других дистрибутивов по важнейшим, но не по всем параметрам.

Управление пользователями

В Linux и в других системах семейства UNIX существует три типа пользователей.

- **Суперпользователь, он же системный администратор, он же root.** Обычно этот пользователь имеет в системе имя root. Если вы войдете в систему под именем root (разумеется, для этого необходимо знать соответствующий пароль), ваши права будут не ограничены: вы можете просматривать, изменять, удалять любые файлы, выполнять любые программы и т. д. Такие широкие полномочия требуются только для администрирования системы. Для выполнения любых других задач наделять пользователя правами администратора из соображений безопасности нельзя.
- **Обычный пользователь.** Таким пользователям система Linux нужна для работы. Они имеют неограниченный доступ к собственным файлам и ограниченный доступ к оставшейся части системы. По возможности их логин должен совпадать с именем (например, kathrin или hofer).
- **Системный пользователь для демонов и служебных программ.** Наконец, существует несколько пользовательских учетных записей, для которых не предусмотрена интерактивная работа с компьютером — такие пользователи нужны для выполнения определенных программ. Например, за работу веб-сервера Apache отвечает не системный администратор, а отдельный пользователь, который в зависимости от дистрибутива может называться apache, wwwrun или httpd и т. д.

  Благодаря наличию таких пользователей гарантируется максимальная безопасность системы.

Файл /etc/passwd

Список всех пользователей сохраняется в файле `/etc/passwd`. Для каждого пользователя указывается следующая информация: логин, полное имя, идентификационные номера пользователя и группы (UID и GID), домашний каталог и используемая оболочка. Формат таков:

Логин:Пароль:UID:GID:Имя:Домашний каталог:Оболочка

В следующих строках записана информация о нескольких пользователях, работающих в Ubuntu Linux (сохранено в каталоге `/etc/passwd`):

```
root:x:0:0:root:/root:/bin/bash
daemon:x:1:1:daemon:/usr/sbin:/bin/sh
bin:x:2:2:bin:/bin:/bin/sh
sys:x:3:3:sys:/dev:/bin/sh
...
```

```
kofler:x:1000:1000:Michael Kofler,,,:/home/kofler:/bin/bash
huber:x:1001:1001:Herbert Huber,,,:/home/huber:/bin/bash
```

По названию `passwd` можно предположить, что в файле также сохраняются пароли. Раньше так и было, но со временем положение изменилось. Вместо паролей `/etc/passwd` теперь содержит только символ `x`. Информация о пароле (которая, разумеется, зашифрована), хранится в отдельном файле `/etc/shadow` (см. подраздел «Пароли» этого раздела).

Логин

Логин (имя для входа в систему) должен состоять только из букв нижнего регистра (буквы и цифры кодировки US-ASCII) и содержать не больше восьми символов. Хотя можно использовать пароли в других кодировках, а также пароли, содержащие более восьми символов, но лучше этого не делать, так как могут возникнуть проблемы совместимости с некоторыми программами. Если отдельно сохранять полные названия, эти ограничения не действуют.

UID и GID

Номер *UID (идентификатор пользователя)* служит для идентификации пользователя. Этот номер сохраняется отдельно в качестве дополнительной информации к каждому файлу, чтобы было известно, кому какой файл принадлежит.

Существуют правила распределения номеров UID: администратор всегда имеет UID = 0. Для серверных служб и демонов в большинстве дистрибутивов предусмотрены идентификационные имена в диапазоне от 1 до 999 (в Red Hat и Fedora, а также в некоторых других дистрибутивах все еще применяются старые правила с использованием диапазона значений от 1 до 499). Для обычных пользователей, соответственно, предусмотрены номера от 1000 (В RHEL 6 и Fedora до версии 15, а также в некоторых других дистрибутивах действовали старые правила: для серверных служб применялся диапазон от 1 до 499, для пользовательских ID — от 500 и выше).

Номер *GID (идентификатор группы)* указывает, к какой группе принадлежит пользователь.

Домашний каталог

Это место, где пользователь может хранить свои личные файлы. Для домашних каталогов обычных пользователей, как правило, применяется путь `/home/login-name`. В домашнем каталоге также сохраняются персональные конфигурационные настройки пользователя и различные программы. Например, в файле с расширением `.emacs` содержатся настройки конфигурации редактора Emacs. Поскольку названия таких конфигурационных файлов обычно начинаются с точки, эти файлы скрыты. Чтобы отобразить такие файлы, введите команду `ls -la`.

Чтобы учетные записи новых пользователей сразу создавались с удобными стандартными настройками, при их создании следует копировать все файлы из `/etc/skel` в новый домашний каталог. Многие программы, создающие новые учетные записи, выполняют эту операцию автоматически. Таким образом, исходные настройки для каждого нового пользователя содержатся в каталоге `/etc/skel`.

Оболочка

Это интерпретатор, с помощью которого пользователь, вошедший в систему, может выполнять команды. Поскольку в Linux на выбор предлагается несколько оболочек, в файле `passwd` необходимо указать, какой оболочкой вы будете пользоваться. В Linux чаще других используется оболочка bash, подробно рассмотренная в главе 6 (в файле `passwd` необходимо сохранять полное название файла оболочки, то есть, например, `/bin/bash`).

Управление группами

Группы требуются для того, чтобы обеспечить нескольким пользователям общий доступ к определенным файлам. Для этого каждый пользователь причисляется к *первичной группе* (initial group). Кроме того, пользователь может относиться к разным *дополнительным группам* (supplementary groups), то есть состоять в нескольких группах сразу.

В файле `/etc/group` хранится полный список групп. Например, в следующих строках показаны определения некоторых групп в файле `/etc/group`. Соблюдается такой формат:

Название_группы:Пароль:Идентификатор_группы:Список_пользователей

Следующие строки взяты из файла `group` системы Ubuntu:

```
root:x:0:
daemon:x:1:
bin:x:2:
...
dialout:x:20:cupsys,kofler,huber
...
users:x:100:
admin:x:114:kofler
...
kofler:x:1000:
huber:x:1001:
...
```

Пользователь соотносится с группой двумя способами.

- Основная группа пользователя сохраняется в `/etc/passwd`. Основная группа пользователя kofler в нашем примере также называется kofler (идентификатор группы 1000 в файле `/etc/passwd`).
- Чтобы задать принадлежность пользователя к другим группам, можно указать его имя в последнем столбце в файле `/etc/group`. Так, kofler принадлежит к группам adm, admin и dialout. Благодаря этому пользователь kofler может выполнять в системах Ubuntu задачи администратора и устанавливать соединение с Интернетом.

При использовании идентификаторов групп номер 0 предусмотрен для администратора, номера с 1 по 99 — для системных служб. Номер GID = 100 обычно

зарезервирован для группы users. Номера GID выше 100 можно распределять по собственному усмотрению.

Соотнесение пользователя с основной группой. Существуют две основные стратегии соотнесения пользователей с их основными группами.

- Устоявшийся метод, используемый в UNIX/Linux уже многие годы, — относить всех обычных пользователей к основной группе users. Этой очень простой стратегии следует, например, SUSE.
- Дистрибутивы, построенные на основе Debian, например Red Hat и Fedora, используют другой подход: каждый пользователь получает собственную основную группу. В таком случае оба пользователя — kofler и huber — будут отнесены к одноименным основным группам. Группа users в итоге больше не нужна.

В некоторых случаях второй подход предпочтительнее — например, когда несколько пользователей дополнительной группы создают общие файлы. Однако этими преимуществами можно воспользоваться лишь при умелом системном администрировании.

В очень многих образцах Linux, где отдельные группы пользователей вообще могут не иметь общих файлов либо необходимо управлять общими проектами, применяются специальные инструменты, например CVS (система контроля параллельных версий). При этом неважно, какие основные группы используются — общая или индивидуальные.

Пароли

Пароли Linux могут состоять только из символов кодировки ASCII (никаких международных специальных символов!). Из соображений безопасности рекомендуется создавать пароли, состоящие из букв как верхнего, так и нижнего регистра; кроме того, пароль должен содержать минимум одну цифру.

Пароли в Linux сохраняются в форме так называемых хеш-кодов, которые позволяют контролировать, но не реконструировать данные. В современных дистрибутивах сейчас используется надежный хеш-алгоритм SHA512 (см. переменную `ENCRYPT_METHOD` в `/etc/login.defs`). Данный алгоритм позволяет создавать пароли любой длины (в некоторых более старых дистрибутивах используются другие хеш-алгоритмы, учитывающие только восемь первых символов в пароле!).

Так называемые взломщики паролей (программы для расшифровки) действуют противоположным образом: они обычно примеряют к программе миллионы случайных паролей — как правило, в таком качестве взломщик использует слова из словаря в комбинации с числами. Поскольку многие пользователи применяют запоминающиеся пароли, а компьютеры становятся все быстрее, подобный метод подбора ужасающе часто приводит злоумышленника к цели.

Чтобы осложнить жизнь потенциальным агрессорам, в новых системах Linux зашифрованные коды-пароли сохраняются не прямо в `/etc/passwd`, а в отдельном файле `/etc/shadow`. Преимущество заключается в том, что этот файл может читать только администратор. (Файлы `/etc/passwd` и `/etc/group` должны быть доступны для чтения всем пользователям системы, так как содержат элементарную информацию

по управлению системой. А к файлу /etc/shadow могут иметь доступ только программы для верификации и изменения паролей. Поэтому потенциальный агрессор должен сначала получить доступ администратора и только потом сможет прочитать зашифрованные коды-пароли.)

Формат файла shadow должен быть следующим:

Логин:Код-пароль:d1:d2:d3:d4:d5:d6:reserved

Далее приведен фрагмент файла:

```
root:$6$Ecrkix...:14391:0:99999:7:::
daemon:*:14391:0:99999:7:::
bin:*:14391:0:99999:7:::
...
kofler:$6$TZR7...:14391:0:99999:7:::
```

Поля d1-d6

В полях d1-d6 могут содержаться желаемые данные по времени:

- d1 указывает, когда пароль был в последний раз изменен (данные указываются в днях, прошедших с 01.01.1970);
- d2 определяет, через сколько дней можно изменить пароль;
- d3 указывает, через сколько дней пароль обязательно необходимо изменить, прежде чем он станет недействителен (подробности об этих полях сообщаются в файле справки man 5 shadow);
- d4 определяет, за сколько дней до истечения срока действия пароля пользователь начнет получать об этом предупреждения;
- d5 указывает, через сколько дней просроченная учетная запись без действующего пароля будет полностью деактивирована;
- d6 задает, когда именно учетная запись будет деактивирована.

Как правило, для d1-d3 применяются стандартные значения, чтобы пароль можно было изменить в любое время и чтобы он оставался действителен в течение неограниченного срока. Но поля d1-d6 можно использовать и для того, чтобы ограничивать сроки действия паролей, автоматически временно деактивировать учетные записи (например, для управления учетными записями студентов в вузе).

Что касается большинства системных пользователей (они рассмотрены выше, например daemon, bin, sys и т. д.), здесь вместо пароля сохраняется только звездочка или восклицательный знак. Это означает, что действующего пароля не существует и войти в систему от имени такого пользователя невозможно. И все же системными учетными записями можно пользоваться: программы, запускаемые с правами администратора, могут потом словно «поменять» своего владельца и продолжать работать уже от имени bin, daemon, lp и т. д. Именно так обстоит дело с большинством системных процессов: они автоматически запускаются от имени администратора при старте системы, а затем сразу же меняют пользователя из соображений безопасности.

Файл /etc/login.defs

Множество параметров, предназначенных для внутрисистемного управления паролями и учетными записями, находится в файле /etc/login.defs. Там можно установить время ожидания после указания неверных данных при входе в систему, количество допустимых попыток входа в систему, количество битов доступа, которые должны использоваться в новых создаваемых домашних каталогах и т. д. Здесь также можно разрешить passwd принимать откровенно слабые пароли.

Изменение паролей

Чтобы изменить собственный пароль, выполните команду passwd. Сначала у вас будет запрошен пароль, а затем вам придется дважды ввести новый пароль. Набираемая информация на экране не отображается. Только если обе последовательности символов для нового пароля совпадут, этот пароль будет принят. Теперь при входе в систему вы должны будете использовать новый пароль.

Длина пароля должна составлять от шести до восьми символов. В паролях могут содержаться буквы верхнего и нижнего регистра, цифры и знаки пунктуации. Команды passwd проводит несколько проверок и отказывается принимать откровенно слабые пароли. В хороших паролях кроме строчных букв содержится минимум одна прописная буква и одна цифра. Кроме того, в качестве паролей нельзя использовать слова (или их небольшие вариации), которые можно найти в словаре.

Обычные пользователи могут изменять лишь собственный пароль, а администратор может изменять и пароли других пользователей.

```
root# passwd hofer
New password: ******
Re-enter new password: ******
Password changed.
```

Забытый пароль администратора

Что делать, если вы забыли пароль администратора? В этом случае нужно загрузить компьютер с установочного диска или восстановительной дискеты либо воспользоваться «живым диском» (например, Knoppix). С помощью mkdir создаем новый каталог на псевдодиске восстановительной или загрузочной системы и привязываем к нему тот сегмент диска, на котором расположена система Linux:

```
root# mount -t ext4 /dev/xxx /dir
```

Теперь у нас есть доступ к файлу dir/etc/shadow, и мы можем удалить в нем пароль администратора (все символы между первым и последним двоеточием) либо заменить его известным нам зашифрованным паролем другого пользователя. Войти в систему с правами администратора, не вводя при этом пароль, можно только из текстовой консоли — такие настройки заданы в файле /etc/pam.d/commonauth из соображений безопасности. Здесь, заново запустив систему, вы устанавливаете новый пароль администратора командой passwd, чтобы можно было работать под X в качестве root.

Если вы хотите не допустить ситуации, в которой любой пользователь мог бы изменить пароль вышеописанным способом, нужно отключить в BIOS компьютера

все загрузочные устройства, кроме первого жесткого диска. Но тогда вам уже никак нельзя забывать пароль. Еще одна возможность — зашифровать целый сегмент диска.

Файл faillog

В зависимости от действующих настроек `/etc/login.defs` информация обо всех неудачных попытках входа в систему определенным образом заносится в `/var/log/faillog`. В отличие от большинства других файлов регистрации, в этом файле используется двоичный формат.

С помощью `faillog -u` *имя* вы определяете, сколько раз заданный пользователь может ошибиться при входе в систему. При превышении этого количества учетная запись блокируется, пока администратор вновь не разблокирует ее командой `faillog -u` *имя* `-r` (команда сбрасывает счетчик попыток).

Кроме того, можно задать единое для всей системы максимальное количество попыток входа в систему с помощью команды `faillog -m max`. Правда, в таком случае вам придется выполнить и `faillog -u root -m 0`, чтобы данная мера не распространялась на администратора. Иначе может случиться так, что вы сами как администратор не сможете войти в систему, после того как другой пользователь несколько раз безуспешно попытается войти в систему с правами администратора.

Групповые пароли

Можно определять пароли не только для отдельных пользователей, но и для целых групп (команда `gpasswd`). Однако в то время как пользовательские пароли необходимы в любом случае, групповые пароли применяются редко. Их основной недостаток заключается в том, что все члены группы должны знать пароль, а это осложняет процесс администрирования.

Если групповые пароли и в самом деле необходимы, их обычно сохраняют в файле `/etc/gshadow`. Групповой пароль нужно вводить в тех случаях, когда пользователь с помощью команды `newgrp` переходит из одной активной группы в другую (активная группа определяет, к какой группе будут относиться новые файлы).

Взаимодействие конфигурационных файлов

Сейчас мы еще раз в обобщенном виде рассмотрим, как взаимодействуют три файла — `passwd`, `groups` и `shadow`. Файл `passwd` каждого пользователя содержит строку, построенную по следующему образцу:

```
# Строка в /etc/passwd
kofler:x:1000:1000:Michael Kofler:/home/kofler:/bin/bash
```

Здесь `kofler` — это логин; `1000` — идентификационный номер пользователя; `1000` — идентификационный номер основной группы; `Michael Kofler` — полное имя (для электронной переписки, новостей и т. д.); `/home/kofler` — домашний каталог пользователя, а `/bin/bash` — используемая им оболочка. Идентификатор группы — это уникальный номер, важный для управления правами доступа к файлам.

Соответствующая строка в `/etc/shadow` с информацией о пароле выглядит следующим образом:

```
# Строка в /etc/shadow
kofler:$6$9dkO$...:13479:0:99999:7:::
```

Последовательность символов после `kofler:` — это зашифрованный пароль. Если не задать эту последовательность, то в систему можно будет входить, не указывая пароль. Если определить вместо нее символ * или !, то вход в систему будет запрещен.

Номер GID из `/etc/passwd` должен соответствовать группе из `/etc/group`. Во многих дистрибутивах каждому обычному пользователю соответствует одноименная группа:

```
# Строка в /etc/group
kofler:x:1000:
```

Управление пользователями в сети

Если вы объединяете в сеть несколько компьютеров с Linux и хотите обеспечить двусторонний доступ к файлам через NFS, не забывайте, что номера UID и GID на всех компьютерах должны быть единообразными. Уже сама организация такого единообразия на множестве компьютеров в сети — задача не из легких. А если вы хотите, чтобы любой пользователь мог войти в систему под своим именем с любого компьютера (разумеется, всякий раз используя для этого одни и те же логин и пароль), то вам понадобится постоянно синхронизировать все файлы `/etc/passwd`. Такое администрирование потребует от вас титанических усилий.

Чтобы облегчить работу, администратор выделяет для управления пользователями специальный центральный сервер. Кроме того, существует множество альтернативных возможностей (протоколов) для аутентификации клиентов, описать которые не позволяют размеры этой книги:

- LDAP — облегченный протокол доступа к сетевым каталогам;
- NIS — сетевая информационная служба, протокол считается устаревшим;
- Kerberos;
- Samba или управление пользователями для Windows.

Подключаемые модули аутентификации (PAM)

Подключаемые модули аутентификации (Pluggable Authentication Modules, PAM) — это библиотека, которая помогает проводить аутентификацию. Если вы входите в систему на компьютере Linux, производите аутентификацию иным образом (`su`, `sudo`, `ssh` и т. д.) или изменяете пароль (`passwd`), то соответствующие программы обращаются к библиотеке PAM. С PAM также работают Cron и PolicyKit. Исчерпывающая, но уже довольно старая документация по PAM имеется на следующем сайте: http://www.kernel.org/pub/linux/libs/pam/.

На обычном компьютере Linux в файле `/etc/passwd` содержатся имена пользователей, а в файле `/etc/shadow` — соответствующие им зашифрованные пароли. По умолчанию PAM конфигурируется так, что может интерпретировать эти данные.

Если требуется дополнительно использовать еще один метод аутентификации (например, LDAP), то конфигурацию PAM необходимо соответствующим образом изменить. Для этого в разных дистрибутивах предназначены различные инструменты:

- Fedora — `system-config-authentication`;
- SUSE — YaST-модуль Безопасность ▸ Управление пользователями и группами;
- Ubuntu — `pam-auth-update`.

Файл pam.conf. Конфигурационные файлы расположены в каталоге `/etc/pam.d/`. Кроме того, интерпретируется файл `/etc/pam.conf`. В конфигурационных файлах построчно перечисляются правила. Каждое правило состоит минимум из трех частей (столбцов):

Тип Реакция PAM-модуль [*аргументы модуля*]

Кроме того, каждой записи в `pam.conf` должно предшествовать имя службы (например, `login` или `other` для стандартных записей). в файлах каталога `/etc/pam.d` каждой службой управляет одноименный файл.

Тип правила (первый столбец). В PAM различается четыре типа правил (типа столбцов). В некоторых дистрибутивах (Debian, Ubuntu) в файлах `common-account`, `common-auth`, `common-password` и `common-session` содержатся стандартные правила для четырех этих типов:

- `account` — позволяет выставлять для служб ограничения, зависящие от времени суток, загруженности, способа входа в систему (например, через консоль) и т. д.;
- `auth` — управляет авторизацией (запросом и проверкой пароля) и дополнительным присвоением привилегий (например, принадлежности к группам);
- `password` — управляет механизмом для изменения пароля;
- `session` — позволяет выполнять действия перед запуском определенной службы или после него (например, журналирование, привязка/отсоединение файловых систем, отображение состояния почтового ящика и т. д.).

Реакция (второй столбец). Следующий столбец в конфигурационных файлах определяет, как должна реагировать библиотека PAM, если правило выполняется или не выполняется. Реакцию можно описать двумя способами: с помощью обычного ключевого слова (например, `required`, `requisite`) либо указав в квадратных скобках пару значение/результат (например `[success=1 new_authtok_reqd=done default=ignore]`). Значение четырех важнейших ключевых слов простого синтаксиса объясняется в табл. 11.3.

Таблица 11.3. Реакция на нарушение правил PAM

Ключевое слово	Реакция
requisite	При нарушении правила функция PAM сразу же возвращает отрицательный результат и остальные правила уже не обрабатываются
required	При нарушении правила функция PAM возвращает отрицательный результат; дальнейшие правила обрабатываются, но результат обработки не учитывается

Ключевое слово	Реакция
sufficient	Если правило выполняется, то функция PAM сразу же возвращает положительный результат (если только уже не было нарушено предшествующее правило requisite); остальные правила не обрабатываются. При нарушении PAM продолжает работу со следующего правила
optional	Результат выполнения правила важен только тогда, когда речь идет о единственном правиле определенного типа и об определенной службе (например, su)

При использовании второго варианта синтаксиса вы указываете несколько пар «значение — результат» в форме `[значение1=результат1 значение2=результат2 ...]`. Для `value` существует множество заданных ключевых слов, выражающих результат выполнения правила; `result` может быть либо числом, указывающим, сколько следующих правил необходимо пропустить, либо ключевым словом, задающим желаемый результат выполнения PAM (`ignore`, `bad`, `die`, `ok`, `done` или `reset`). В следующем листинге показано, как выразить ключевые слова первого варианта синтаксиса вторым способом записи:

```
requisite  = [success=ok new_authtok_reqd=ok ignore=ignore default=die]
required   = [success=ok new_authtok_reqd=ok ignore=ignore default=bad]
sufficient = [success=done new_authtok_reqd=done default=ignore]
optional   = [success=ok new_authtok_reqd=ok default=ignore]
```

Модуль и параметры (третий столбец). В третьем столбце указывается номер модуля PAM, который будет применяться для интерпретации правила. На работу модуля могут влиять параметры. К сожалению, не существует общего документа о том, какие параметры допустимы, и об их значении.

В традиционной конфигурации модуль `pam_unix.so` отвечает за интерпретацию файлов `/etc/passwd` и `/etc/shadow`. Параметр `nullok` позволяет вводить пустые пароли; `nullok_secure` разрешает аутентификацию с пустым паролем, если вход в систему происходит через консоль, указанную в `/etc/securetty` (по умолчанию в этом файле перечисляются локальные консоли `/dev/ttyn`). С параметром `obscure` модуль проводит несколько простейших тестов, чтобы определить, не слишком ли тривиален пароль. Параметр `md5` указывает, сколько паролей необходимо зашифровать.

Стандартная конфигурация. В следующем листинге обобщены стандартные настройки по всем четырем типам правил в конфигурации Fedora 17. Обратите внимание на то, что стандартные настройки от дистрибутива к дистрибутиву значительно различаются и часто распределены между несколькими файлами (например, `common-xxx` в Debian и Ubuntu).

```
# Файл /etc/pam.d/password-auth (Fedora)
auth     required  pam_env.so
auth     sufficient   pam_unix.so nullok try_first_pass
auth     requisite pam_succeed_if.so uid >= 1000 quiet_success
auth     required  pam_deny.so

account  required  pam_unix.so
account  sufficient   pam_localuser.so
account  sufficient   pam_succeed_if.so uid < 1000 quiet
account  required  pam_permit.so
```

```
password  requisite pam_pwquality.so try_first_pass retry=3 type=
password  sufficient   pam_unix.so sha512 shadow nullok try_first_pass use_authtok
password  required  pam_deny.so

session   optional  pam_keyinit.so revoke
session   required  pam_limits.so
session   [success=1 default=ignore] pam_succeed_if.so service in crond quiet use_uid
session   required  pam_unix.so
```

Диспетчер переключения имен (NSS)

При входе в систему, управлении правами доступа к файлам, доступе к сети и т. д. может потребоваться практически любая информация об именах пользователей и групп, UID и GID, хост-именах, портах сетевых служб и т. д. По умолчанию эти данные находятся в файлах `/etc/passwd`, `/etc/group`, `/etc/hosts`, `/etc/services` и т. д.

В номенклатуре UNIX или Linux доступ к таким службам обобщается термином *службы имен*. За выполнение этих задач отвечает диспетчер службы имен (NSS), сборник функций библиотеки `libc`. Как и при аутентификации, можно настроить источник данных для NSS, например, в том случае, если данные предоставляет сервер LDAP. Важнейшим конфигурационным файлом является `/etc/nsswitch.conf`. По умолчанию в Ubuntu 12.10 этот файл содержит следующие строки:

```
# Файл /etc/nsswitch.conf (Ubuntu)
passwd:      compat
group:       compat
shadow:      compat
hosts:       files mdns4_minimal [NOTFOUND=return] dns mdns4
networks:    files
protocols:   db files
...
```

Первый столбец в `nsswitch.conf` содержит информацию о базе данных или файле. После двоеточия стоят ключевые слова, описывающие способ доступа к данным, а также иногда реакцию на полученные данные (в квадратных скобках). В следующем списке объяснены важнейшие ключевые слова, связанные с методами доступа. Если в одной строке указано несколько методов доступа, они применяются по порядку, пока один из них не подойдет. Более подробная информация по синтаксису содержится в `man nsswitch.conf`.

- `files` — обращается к традиционным конфигурационным файлам, например `/etc/passwd` или `/etc/group`.
- `compat` — аналогично `files`, причем данным о пользователях и группах могут предшествовать символы `+` и `-`. Так улучшается совместимость с NIS. Это слово нельзя комбинировать с другими ключевыми словами.
- `db` — считывает информацию из файла баз данных BDB (Berkeley Database).
- `dns` — связывается с сервером имен.
- `mdns` — использует многоадресную DNS (она же Zeroconf или Apple Bonjour/Rendezvous).

Для работы методов доступа необходимо установить надлежащую библиотеку, например `libnss_db` для `db`. Если библиотека не установлена, то соответствующее ключевое слово просто игнорируется (без сообщения об ошибке).

Nscd (демон кэширования службы имен)

В некоторых дистрибутивах (например, Fedora, Red Hat и SUSE) по умолчанию устанавливается программа `nscd`, активируемая при выполнении процесса Init-V. В Debian и Ubuntu ее также можно установить.

Аббревиатура nscd означает Name Service Caching Daemon (демон кэширования службы имен). Эта программа отмечает логины, имена групп и хост-имена, применяемые при конфигурации системы, а также их IP-номера. В отличие от сервера имен, `nscd` предоставляет эту информацию только локальному компьютеру, но не остальным компьютерам, подключенным к сети. Использовать `nscd` целесообразно лишь тогда, когда управление пользователями происходит через сетевую службу (например, LDAP). В таком случае `nscd` действует в качестве кэша для информации, исключая избыточные запросы LDAP, ответы на которые иногда приходят достаточно нескоро.

Файл /etc/nscd.conf. Конфигурация `nscd` осуществляется в `/etc/nscd.conf`. Обычно записи в этом файле подразделяются на три группы: `passwd` для логинов, `group` для групп и `host` для хост-имен. Настройки для каждой группы определяют, как долго должны храниться файлы, максимальное количество записей, которыми необходимо управлять, и т. д.

11.5. Языковые настройки, интернационализация, Unicode

В этом разделе будут рассмотрены два вопроса.

- **Локализация или языковые настройки** — определяет, на каком языке будут отображаться сообщения об ошибках, меню, диалоговые окна, тексты справки и т. д. В соответствии с этими настройками также изменяется формат даты, времени, валюты и т. п.
- **Кодировка** — задает, какие коды используются для сохранения букв. В данном случае общепризнанные правила существуют только для 7-битной ASCII. Однако практически во всех кодировках букве a соответствует код 65. Существуют отклонения, касающиеся международных символов. Например, буква Ä может соответствовать различным кодам, в зависимости от применяемой кодировки.

При работе с локализацией программ вы будете часто сталкиваться с загадочными сокращениями i18n и l10n. Они означают, соответственно, *интернационализацию* (internationalization = i + 18 букв + n) или *локализацию* (localization = l + + 10 букв + n). Такие сокращения хороши при работе с поисковиками, если вам понадобится дополнительная информация по этим темам.

Основы кодировок

Кодировка (character set) описывает соответствия между числовыми кодами и буквами. Наиболее известны кодировки ASCII (7 бит), ISO-латиница-n (8 бит) и Unicode (16 бит).

- **ASCII** — включает только 127 символов, в том числе буквы от a до z (или от A до Z), цифры от 0 до 9 и различные знаки пунктуации.
- **ISO-8859, латиница** — кодировки ISO содержат, кроме 127 символов ASCII, до 128 дополнительных символов для различных языковых регионов. В следующем списке перечислены важнейшие кодировки ISO:
  - ISO-8859-1 = «латиница-1» — западноевропейская;
  - ISO-8859-2 = «латиница-2» — центрально- и восточноевропейская;
  - ISO-8859-3 = «латиница-3» — южноевропейская;
  - ISO-8859-4 = «латиница-4» — северноевропейская;
  - ISO-8859-9 = «латиница-5» — турецкая;
  - ISO-8859-10 = «латиница-6» — северная (саамская, эскимосская, исландская);
  - ISO-8859-13 = «латиница-7» — балтийская;
  - ISO-8859-14 = «латиница-8» — кельтская;
  - ISO-8859-15 = «латиница-9» = «латиница-0» — сходна с «латиница-1», но с символом евро.

  «Латиница-0» пока имеет статус проекта и не получила официального названия, но используется достаточно часто. В Windows кодировки называются *кодовыми страницами* (code pages). Code Page 1252 очень похожа на «латиницу-1», но есть и некоторые отличия.
- **Unicode** — чтобы разрешить путаницу различных 8-битных кодировок, была разработана 16-битная кодировка Unicode, она же ISO-10646. Ею можно закодировать не только все символы европейских языков, но и большинство азиатских символов (поскольку для каждого символа предусмотрено 16 бит, в этой кодировке найдется место для более чем 65 000 символов).

  Unicode определяет, какой код будет присвоен какому символу, но не задает метод сохранения кодов. На первый взгляд кажется, что проще всего было бы представить каждый символ в виде 2 байт (то есть 16 бит). Такой формат называется UTF-16 (Unicode Transfer Format — формат передачи Unicode). Но у него есть два недостатка: во-первых, удваивается потребность в памяти, даже в тех случаях, когда требуется сохранять преимущественно европейские символы или даже только символы US-ASCII. Во-вторых, байтовый код 0 оказывается где угодно в последовательностях символов Unicode. Но многие программы C, почтовые серверы и т. д. настроены так, что байт 0 означает конец последовательности символов.

  Есть и другие возможности представления текстов в Unicode. Популярнейшей альтернативой UTF-16 является UTF-8. При этом символы US-ASCII (7 бит), как и ранее, представлены одним байтом, наибольший разряд в котором равен 0.

Все остальные символы представляются в виде последовательностей длиной от 2 до 4 байт. Очевидный недостаток этого формата заключается в том, что отсутствует непосредственная взаимосвязь между размером документа в байтах и количеством символов в документе. Однако поскольку UTF-8 лучше совместима с существующими программами и обладает другими достоинствами, в UNIX/Linux она считается стандартом. В Microsoft Windows часто используется UTF-16. Итак, если мы говорим о Unicode и имеем в виду работу с Linux, то, как правило, речь идет об UTF-8. Практически во всех дистрибутивах UTF-8 применяется по умолчанию.

Влияние кодировки

Действующая кодировка определяет, как именно будут кодироваться символы в текстовых файлах или в именах файлов. Файловые системы Linux понимают любую кодировку. Именем файла считается всякая последовательность символов, которая заканчивается байт-кодом 0. Но в зависимости от того, какая кодировка сейчас используется, последовательность и количество байт для такого имени файла, как `äöü.txt`, может быть совершенно разной! Если применяемая кодировка — «латиница-1», то это имя можно выразить 7 байтами (плюс один нулевой байт). Если же используется кодировка Unicode/UTF-8, то имя файла будет иметь длину 10 байт (так как для представления каждого из символов — ä, ö, ü — требуется по 3 байта).

Множество программ функционируют независимо от кодировки либо могут понимать сразу несколько кодировок: например, почтовые клиенты и браузеры могут отображать письма и сайты, где используется не та кодировка, которая сейчас задана в качестве активной. Современные программы для обработки текста обычно сохраняют текст в Unicode либо задействуют собственный код. Такие редакторы, как Emacs или XEmacs, обычно также могут обрабатывать и сохранять текстовые файлы в разных кодировках.

Проблемы с кодировками

Чаще всего проблемы возникают, если отправитель и получатель текстового файла применяют разные кодировки. Например, пользователь, работающий с дистрибутивом Linux, задействует кодировку Unicode и создает в текстовом редакторе файл, содержащий международные специальные символы. И когда пользователю, работающему с другой оперативной системой, где установлена кодировка-латиница, придется продолжить обработку этого файла, он с удивлением обнаружит, что символы, не относящиеся к ASCII, отображаются неправильно. Обычно такие проблемы легко решаются с помощью команд `recode` или `iconf` (см. раздел 9.3).

Аналогичные проблемы касаются имен файлов, в частности, при работе с NFS3: если вы создаете на компьютере, использующем кодировку UTF8, файл `äöü.txt`, а потом обращаетесь к этому файлу с другого компьютера с кодировкой-латиницей через NFS, то имя файла будет выглядеть примерно как `Ã¤Ã¶Ã¼.txt`. Эта проблема решается применением одинаковой кодировки на всех компьютерах в сети или использованием NFS4. Если вы хотите изменить кодировку для имен уже имеющихся файлов (таких файлов может быть очень много), то вам поможет команда `convmv`, описанная в разделе 9.4.

Шрифт (гарнитура)

Не следует путать шрифт с кодировкой. Шрифт отвечает за то, как именно определенный символ будет отображаться на экране. Для этого существуют различные гарнитуры. Arial, Courier, Helvetica, Palatino — наиболее известные из них.

Разумеется, между шрифтами и кодировкой есть некоторое сходство: чтобы символ с кодом 234 мог правильно отображаться на экране, должно быть известно, какой набор символов использовался при кодировке. Некоторые старые X-шрифты были ограничены 256 символами и существовали в виде нескольких отдельных версий для различных кодировок. Новые гарнитуры (TrueType, PostScript) содержат гораздо больше символов и совместимы с несколькими кодировками.

Настройка локализации и кодировки

В зависимости от дистрибутива или операционной системы домашнего компьютера для конфигурирования языковых настроек применяются различные инструменты. Практически в любых случаях задействуется кодировка UTF-8. Лишь в немногих дистрибутивах сохранилась возможность использования 8-битной кодировки. Во всех дистрибутивах следует заново войти в систему, чтобы измененные языковые настройки вступили в силу. Gnome учитывает языковые настройки системы, но сам не содержит никаких конфигурационных инструментов.

- Debian — `dpkg-reconfigure locales`;
- Fedora, Red Hat — `system-config-language`;
- Gnome 3.*n* — системные настройки Регион и язык;
- KDE — модуль центра управления Персональные данные ▸ Страна/Регион;
- SUSE — YaST-модуль Система ▸ Язык;
- Ubuntu — `gnome-language-selector`.

Кроме того, в диалоговых окнах входа в систему в KDE и Gnome (то есть в `kdm` и `gdm`) существует возможность выбрать для следующего соединения желаемый язык.

Конфигурационные файлы

Неудивительно, что в разных дистрибутивах конфигурационные настройки сохраняются в различных местах:

- Debian, Ubuntu — `/etc/default/locale`;
- Red Hat, Fedora — `/etc/sysconfig/i18n`;
- SUSE — `/etc/sysconfig/language`.

Кроме того, во многих дистрибутивах учитываются пользовательские настройки, сохраняемые в файле `~/.i18n`.

Переменные LC и LANG

Внутри системы локализация и кодировка управляются переменными окружения `LC_CTYPE` и `LANG`. Для интерпретации этих переменных применяется библиотека `glibc`, используемая почти во всех программах Linux.

Локализация может проводиться категориально. При правильной конфигурации можно, например, применять немецкий формат для указания даты и времени, а сообщения об ошибках отображать по-английски. В табл. 11.4 перечислены важнейшие переменные.

Таблица 11.4. Важнейшие локализационные переменные

Переменная	Значение
LANG	Определяет стандартное значение для всех ненастроенных LC-переменных
LC_CTYPE	Задает кодировку
LC_COLLATE	Указывает порядок сортировки
LC_MESSAGES	Определяет вид уведомлений, сообщений об ошибках и т. д.
LC_NUMERIC	Задает представление чисел
LC_TIME	Определяет представление даты и времени
LC_MONETARY	Задает представление денежных сумм
LC_PAPER	Определяет размер бумаги
LC_ALL	Имеет приоритет над всеми индивидуальными LC-настройками

Разумеется, не в каждой программе учитываются все эти категории (а во многих программах переменные LC_ полностью игнорируются). Если отдельные категории не настроены, то программы применяют в качестве стандартного значения C или POSIX. Это означает, что сообщения об ошибках выводятся по-английски, дата и время отображаются в американском формате и т. д.

Чтобы не настраивать по отдельности все перечисленные здесь переменные, можно просто задать переменную LANG. При этом для всех переменных, не определенных особо, будет применяться стандартное значение LANG. Только переменная LC_COLLATE сохранит базовую настройку POSIX. В большинстве дистрибутивов языковые настройки полностью устанавливаются через переменную LANG.

Переменная LC_ALL мощнее, чем LANG. Если установить LC_ALL, ее настройки будут действовать для всех категорий (независимо от того, как настроены другие переменные LC или LANG).

В большинстве программ сообщения об ошибках и другие тексты располагаются в отдельных каталогах, например /usr/share/locale*/*язык*/LC_MESSAGES. Другая необходимая информация по темам локализации и интернационализации находится на странице man о команде locale и на веб-странице http://www.gnu.org/software/libc/manual/.

Тестирование локализации

Текущее состояние настроек локализации проще всего определить с помощью команды locale. Она также интерпретирует LANG и LC_ALL и узнает результирующие параметры. В следующем примере показаны настройки, действующие на моем компьютере:

```
user$ locale
LANG=de_DE.UTF-8
LC_CTYPE="de_AT.UTF-8"
LC_NUMERIC="de_AT.UTF-8"
LC_TIME="de_AT.UTF-8"
```

```
...
LC_ALL=
```

Чтобы протестировать локализацию, можно также выполнить с ошибкой любую команду. Сообщение об ошибке должно быть выведено на том языке, который был настроен. Если настройкой LANG является de_DE, то сообщение об ошибке выполнения команды mount должно выглядеть как в следующем примере.

```
user$ mount /xy
mount: Konnte /xy nicht in /etc/fstab oder /etc/mtab finden
```

Команда env

Если вы хотите выполнить отдельную команду с применением настроек другого языка, не изменяя всю конфигурацию, то лучше всего воспользоваться командой env. Она ожидает присвоения отдельных переменных и, наконец, самой команды, которая должна быть выполнена с учетом настроенных переменных:

```
user$ env LANG=C mount /xy
mount: can't find /xy in /etc/fstab or /etc/mtab
```

Если настройка LANG изменена, а сообщение об ошибке все еще выводится не по-английски, попробуйте также сбросить настройки LANGUAGE:

```
user$ env LANG=C LANGUAGE=C mount /xy
mount: can't find /xy in /etc/fstab or /etc/mtab
```

Чтобы настроить LANG для всей сессии целиком, выполните команду export LANG=C.

Допустимые настройки LC/LANG

Список всех возможных настроек для переменных можно узнать с помощью команды locale -a. Обычно используется запись вида *x_y*, причем *x*, записываемый двумя буквами, означает язык, а *y*, также записываемая двумя буквами, означает страну. В немецкоязычном регионе следует использовать настройки de_DE. Английская стандартная настройка может записываться кратко — C.

Новые версии glibc понимают и такие настройки, как deutsch или german. В файле /usr/share/locale/locale.alias содержится таблица, в которую занесены допустимые сокращенные записи и соответствующие им полные локализационные названия.

Пакеты локализации

Чтобы меню, диалоговые окна, сообщения об ошибках, тексты справки и т. д. отображались на нужном языке, следует установить необходимые пакеты локализации. Если вы хотите настроить для своего дистрибутива, например, русский язык, вам следует установить соответствующие дополнительные пакеты для Gnome, KDE, OpenOffice, Firefox и т. д. При работе со SUSE и Ubuntu вам помогут конфигурационные инструменты, перечисленные в начале главы, а в других дистрибутивах установка выполняется вручную.

Не каждая программа Linux локализована для всех языков. Особенно недостает переводов электронной документации (страниц man, пособий, справочных систем).

Если подходящих файлов локализации нет, то Linux отображает информацию по-английски.

Настройка кодировки. Вместе с локализацией настраивается и кодировка. Она указывается после кода страны, отделяется от этого кода точкой, например `de_DE.ISO-8859-1` или `de_DE.utf8`.

11.6. Справка по аппаратным компонентам

В этой книге нет отдельной главы, посвященной аппаратному обеспечению. Однако правильная конфигурация оборудования рассматривается в тематически родственных главах. Иначе говоря, если у вас возникнут проблемы с сетевой картой, почитайте об этом в главе 18, где рассматривается конфигурация сети.

Итак, в этом разделе решаются две задачи: во-первых, систематизированная справка должна облегчить вам поиск дальнейшей информации об определенных аппаратных компонентах. Во-вторых, дается краткая информация об оборудовании, которое будет упоминаться в оставшейся части книги.

Разумеется, есть множество аппаратных компонентов, которые по различным причинам *не получится* описать в этой книге. Просто у меня нет возможности провести все те разнообразные тесты, которые может себе позволить, например, компьютерный журнал.

СОВЕТ

Перед покупкой оборудования узнайте, совместим ли приобретаемый вами компонент с Linux! Посмотрите сайты по аппаратному обеспечению в Linux. Попробуйте поискать в Интернете по запросу вида «linux название модели». Эти вопросы, естественно, обсуждаются в специальных журналах, ориентированных на Linux, и информация в них актуальнее, чем в книгах.

Файлы-устройства. Основные аппаратные компоненты запрашиваются через так называемые файлы-устройства, например `/dev/sda` для жесткого диска SATA. При этом файлы-устройства динамически создаются системой udev. Список важнейших файлов-устройств Linux приводится в табл. 7.8 раздела 7.9.

Proc- и sys-файлы. Файловые системы дают подробную информацию по многим аппаратным компонентам в каталогах `/proc` и `/sys`..

Модули ядра. Драйверы к многочисленным аппаратным компонентам находятся в модулях ядра. Часть этих модулей загружается при запуске системы, а оставшиеся — только при необходимости. Если автоматическая загрузка модулей не работает, посмотрите файлы `/etc/mmodprobe.conf` и `/etc/modprobe.conf.d/*`.

Обзор аппаратного обеспечения. Чтобы получить обзор действующего аппаратного обеспечения, выполните команды `lshal`, `lspci` и `lsusb`. Кроме того, не помешает просмотреть сообщения ядра с помощью команды `dmesg`.

Процессор и память

Процессор. В этой книге рассматриваются только однопроцессорные компьютеры, совместимые с Intel-Pentium, то есть 32-битные и 64-битные процессоры Intel

и AMD. Есть еще и версии Linux для процессоров со всевозможными иными вариантами архитектуры (например, для PowerPC).

О том, какие процессоры работают у вас на компьютере, вы можете узнать, просмотрев файл /proc/cpuinfo. Следующий, сильно сокращенный вывод получен при проверке компьютера с процессором Intel-i7. Linux воспринимает оба ядра как самостоятельные процессоры. При этом в строке model name указывается максимальная тактовая частота, а в строке cpu MHz — текущая тактовая частота.

```
user$ cat /proc/cpuinfo
cat /proc/cpuinfo
processor    : 0
model name   : Intel(R) Core(TM) i7 CPU      860@ 2.80GHz
...
processor    : 1
model name   : Intel(R) Core(TM) i7 CPU      860@ 2.80GHz
...
```

В процессорах с переменной тактовой частотой предусмотрены модули cpufreq, отвечающие за временное снижение частоты (в целях энергосбережения). Информация по этой системе содержится на следующих сайтах: http://www.kernel.org/doc/Documentation/cpu-freq/; https://wiki.archlinux.org/index.php/CPU_Frequency_Scaling.

Сведения о текущем состоянии системы и о возможностях управления находятся в файлах следующего каталога: /sys/devices/system/cpu/cpu*n*/cpufreq/.

Ограничение рабочей частоты процессора. Чтобы при работе процессор не нагревался сильно, можно ограничивать максимальную частоту процессора. Для этого применяется команда cpufreq-set из пакета cpufrequtils. Следующая команда ограничивает частоту значением 800 MHz:

```
root# cpufreq-set -r -max 0.8GHz
```

Полезность этой команды вызывает споры: задачи, требующие большой вычислительной мощности, выполняются дольше, а при этом, как правило, выделяется еще больше тепла.

Отслеживание температуры процессора. Если вы хотите узнать, какова в данный момент температура процессора, установите пакет lm-sensors. После установки выполните от имени root команду sensors-detect. Она определяет, о каких аппаратных компонентах в системе предоставляется информация. Кроме процессора, такие сведения можно узнать о жестком диске, графической карте или различных вентиляторах, которые сообщают свою скорость вращения. Модули ядра, необходимые для такой оценки, можно по желанию вставить в /etc/modules с помощью sensors-detect. Но такие настройки вступают в силу только после перезапуска системы. Чтобы обойтись без перезапуска, можно вручную загрузить интересующий вас модуль с помощью modprobe.

После этой подготовительной работы команда sensors выдает список значений текущих температур:

```
root# sensors
coretemp-isa-0000
Adapter: ISA adapter
```

```
Physical id 0:  +31.0°C      (high = +80.0°C, crit = +98.0°C)
Core 0:         +26.0°C      (high = +80.0°C, crit = +98.0°C)
Core 1:         +29.0°C      (high = +80.0°C, crit = +98.0°C)
```

Есть программы, представляющие эти данные в более удобном виде. Можете поэкспериментировать с программой `xsensors`, которая отображает данные сенсоров в специальном окне. Чтобы отображать эту информацию в Ubuntu на панельном индикаторе, установите пакет `indicator-sensors`.

Оперативная память (RAM). Информация об имеющейся в распоряжении памяти выдается при введении команды `free`. Если вам кажется, что у компьютера возникли аппаратные проблемы с оперативной памятью (неисправные модули памяти), воспользуйтесь программой Memtest86 — это отличный инструмент для тестирования памяти. Если она у вас не заработает, скачайте с сайта http://www.memtest86.com/ ISO-образ, чтобы записать загрузочный диск.

Управление энергопотреблением

ACPI. Аббревиатура ACPI означает Advanced Configuration and Power Interface (улучшенный интерфейс для конфигурации и управления электропитанием). Этот интерфейс предназначен для управления функциями энергопотребления в имеющихся на рынке ПК и ноутбуках и используется примерно с 1999 года. Ранее применялся стандарт APM (автоматическое управление питанием). ACPI поддерживается в Linux, начиная с версии ядра 2.6. Необходимые модули ядра загружаются автоматически — можете убедиться в этом, выполнив команду `dmesg | grep ACPI`. Одновременно запускается процесс ядра `kacpid` и демон ACPI `acpid`. Обе программы обрабатывают события ACPI. Демон `acpid` управляется файлами, находящимися в каталоге `/etc/acpi`.

Команда `acpi -V` и файлы каталога `/proc/acpi` содержат информацию о текущем состоянии системы ACPI (загруженность батареи, ее температура и т. д.).

Если ACPI вызывает проблемы при запуске, обратитесь к подразделу «Параметры ACPI» раздела 17.4, где описаны некоторые параметры ядра, позволяющие полностью или частично отключить ACPI. Прочая информация по ACPI и Linux содержится по адресу http://www.lesswatts.org/projects/acpi/.

Suspend. В ACPI поддерживается несколько разновидностей спящего режима, при которых компьютер потребляет мало энергии (режим Stand-by) либо вообще не потребляет ее (сон, гибернация, ждущий режим). В большинстве дистрибутивов и локальных систем компьютер переводится в нужный спящий режим с помощью команд меню, например Система ▸ Выключение.

При работе с ноутбуком наиболее интересен режим гибернации. При его включении содержимое оперативной памяти записывается в специальный своп-раздел на жестком диске и компьютер полностью выключается. Предполагается, что такой раздел должен быть достаточно велик!

При активации данные оперативной памяти снова считываются с жесткого диска. Кроме того, заново инициализируются все аппаратные компоненты. Этот процесс очень сложен и требует безукоризненного взаимодействия ядра Linux, его модулей и системы ACPI.

ВНИМАНИЕ

Предупреждаю, что мой опыт работы с различными спящими режимами был, в основном, горьким. Если при попытке активации в системе возникала какая-нибудь ошибка, вывести компьютер из спящего режима уже не получалось.

С несколько большим успехом мне удавалось переводить тестовые компьютеры в режим ожидания, который надежно работал практически на всех устройствах.

В любом случае рекомендую приступать к тестированию функции Suspend с осторожностью: сначала сохраните все важные данные, выполните sync и отключите от дерева каталогов все файловые системы, не нужные в настоящий момент.

Меры по экономии энергии. В последние годы в сообществе разработчиков Linux прилагается масса усилий для того, чтобы минимизировать затраты энергии и увеличить длительность работы ноутбука на аккумуляторе. Хороший обзор распространенных методов и соответствующих инструментов дается на следующем сайте: http://www.lesswatts.org/projects/.

Многие операции, описанные на этой странице, уже вошли в состав ядра, стали стандартными настройками современных дистрибутивов и включены в инструменты управления энергопотреблением KDE и Gnome.

Powertop. Команда `powertop` полезна при поиске программ, выводящих процессор из состояния бездействия (idle). Одновременно программа сообщает, как можно минимизировать потребление энергии.

```
root#  powertop
Cn                       Задержка        P-States     (Частоты)
C0 (Процессор работает)  ( 1,8%)         1,80 GHz     0,0%
циклический AbfraC1 остановка     0,0m   1,60 GHz     0,0%
C1 остановка    0,0ms    ( 0,0%)         1400 MHz     0,0%
C2              17,7ms   (98,2%)         1200 MHz     0,0%
C3              0,0ms    ( 0,0%)         600 MHz      100,0%
C4              0,0ms    ( 0,0%)
Активаций в секунду: 55,3     Интервал: 3,0s
Энергопотребление (Оценка ACPI): 20,1W (1,2 Std.)
Наиболее частые причины активации:
23,0% (15,7)   <interrupt>       : ehci_hcd:usb1, uhci_hcd:usb2, uhci_hcd:usb3,
21,6% (14,7)  USB-устройство 3-2 : Optical USB Mouse (Logitech)
13,7% ( 9,3)         <interrupt> : extra timer interrupt
12,7% ( 8,7)       <Kernel Kern> : usb_hcd_poll_rh_status (rh_timer_func)
 9,8% ( 6,7)       <Kernel Kern> : hrtimer_start (tick_sched_timer)
 3,9% ( 2,7)      gnome-terminal : schedule_hrtimeout_range (hrtimer_wakeup)
Предложение: Активируйте "USB autosuspend", нажав клавишу U, либо задайте параметр
загрузки "usbcore.autosuspend=1" в командной строке ядра, либо укажите этот параметр
в конфигурации GRUB.
```

Инструмент `powertop` — это то, что вам нужно, если вы специально хотите найти способы, которые помогли бы продлить срок службы вашего ноутбука.

Режим ноутбука. Это функция ядра, призванная свести к минимуму энергопотребление ноутбука, работающего от аккумуляторной батареи. Основная функция режима заключается в том, чтобы осуществлять запись данных на диск не сразу, а сохраняя в кэш. Синхронизация данных происходит лишь с интервалом в не-

сколько минут (если только не требуется одновременно сохранить очень много данных). В периоды между сохранениями информации жесткий диск можно перевести в энергосберегающий режим. Этот режим активизируется следующей командой:

```
root# echo 5 > /proc/sys/vm/laptop_mode
```

Обратите внимание, если часто включать и выключать машину, срок службы жесткого диска может снизиться (см. также http://lwn.net/Articles/257426/). Не используйте режим ноутбука при работе с ПК, так как винчестеры ПК обычно рассчитаны на менее частое включение и выключение. Кроме того, необходимо сознавать, что если электричество отключится, вы потеряете все несохраненные данные.

Как правило, режим ноутбука управляется сценариями и конфигурационными файлами пакета `laptop-mode-tools`. Кроме того, этот пакет позволяет проводить и многие другие операции по сбережению энергии. Конфигурация осуществляется в каталоге `/etc/laptop-mode` (см. `laptop-mode.conf`). Дополнительную информацию по режиму ноутбука и инструментам `laptop-mode-tools` вы найдете по адресу http://samwel.tk/laptop_mode/.

Управление системой вентиляции

На мой взгляд, нет ничего более надоедливого, чем тихий ноутбук, в котором постоянно воет вентилятор. Можно ли с этим что-нибудь поделать? Уже при покупке ноутбука старайтесь выбрать модель, которая не очень шумит. Но тем, кто хочет уменьшить шум с помощью программных средств, все-таки можно кое-что посоветовать.

Управление вентилятором. В некоторых ноутбуках есть специальные программы для управления вентилятором. Для работы с ними нужно предварительно установить и конфигурировать пакет `lm-sensors`. Это необходимо, чтобы можно было отслеживать температуру процессора. Если данный пакет установлен и подготовлен к работе, вы можете указать, при какой температуре должен срабатывать какой вентилятор и с какой скоростью он при этом должен вращаться.

Одна из самых популярных программ для управления вентилятором называется `tpfanco`. Она совместима практически со всеми ноутбуками IBM и Lenovo. Актуальные пакеты для Debian можно скачать по следующему адресу: http://code.google.com/p/tpfanco/downloads/list.

После того как вы установите и вручную запустите демон `tpfan` (`service tpfand start`), можно будет управлять вентиляторами с помощью `tpfan-admin` (но для этого нужны права администратора). В графическом пользовательском интерфейсе можно отдельно настроить порог срабатывания для каждого температурного сенсора, распознанного в системе.

Я тестировал эту программу на ноутбуке Lenovo E320 в течение нескольких дней. Мне удалось добиться того, что вентилятор стал вращаться менее активно, но полностью заглушить его я не смог.

ВНИМАНИЕ

Разумеется, у ручного управления вентиляторами есть негативные стороны. Если процессор или другие компоненты компьютера регулярно перегреваются, то они изнашиваются быстрее! Поэтому используйте программы для управления вентилятором с осторожностью и сначала почитайте в Интернете, какой опыт имели с этими приложениями другие пользователи.

Интерфейсы и системы шин

Последовательные и параллельные интерфейсы. В Linux последовательные и параллельные интерфейсы доступны через файлы-устройства `/dev/ttySn` или `/dev/lpn`. Скорее всего, вы сталкивались с такими устаревшими интерфейсами при конфигурации аналогового модема или старого принтера.

IDE, SATA, SCSI. Внутренние жесткие диски, приводы CD или DVD, а также многие другие носители данных, как правило, подсоединяются к компьютеру через системы шин IDE, SATA или SCSI. Современные версии Linux сообщаются с устройствами IDE, SATA или SCSI через SCSI-систему ядра. Лишь некоторые DIE-контроллеры, несовместимые с расширением libata системы SCSI, все еще используют старые драйверы IDE.

Информация о состоянии систем IDE и SCSI и всех связанных с ними устройств выводится командой `lsscsi`, а также содержится в следующих файлах:

- `/sys/bus/ide/*`;
- `/sys/bus/scsi/*`;
- `/proc/scsi/*`.

USB. *Универсальная последовательная шина* используется для связи компьютера с самыми разными внешними устройствами: от мыши до сканера. Необходимые для работы USB модули ядра загружаются автоматически. USB-носители (внешние винчестеры, флешки, внешние CD- и DVD-приводы) воспринимаются системой как SCSI-устройства.

Виртуальная файловая система usbfs сообщает информацию обо всех подключенных USB-устройствах в каталоге `/proc/usb`. Затем можно обратиться к данным, содержащимся в каталоге `/sys/bus/usb`. Подробный список всех USB-интерфейсов и устройств возвращает команда `lsusb -v` (пакет `usbutils`).

Firewire. Система шин Firewire — альтернатива USB. Firewire определяется стандартом IEEE 1394 и известна также под названием i.Link, используемым фирмой Sony. Firewire действует несколько быстрее, чем USB, и особенно популярна для передачи данных с видеокамеры. Подробная информация о IEEE 1394 и связи этого стандарта с Linux находится на сайте http://www.linux1394.org/.

При подсоединении устройств Firewire автоматически загружаются требуемые модули, в частности, ieee1394. Информация о подсоединенных устройствах и о состоянии системы Firewire содержится в файлах каталога `/sys/bus/ieee1394`.

PCI. Информацию о компонентах PCI, установленных на вашем компьютере, можно узнать, введя команду `lspci`. Файлы в каталогах `/proc/bus/pci/` и `/sys/bus/pci/` содержат ту же информацию, но они гораздо менее удобны для интерпретации. Следующий вывод ради экономии места сильно сокращен:

```
root# lspci
00:00.0 Host bridge: Intel Corporation 82P965/G965 Memory Controller Hub
00:01.0 PCI bridge: Intel Corporation 82P965/G965 PCI Express Root Port
00:1a.0 USB Controller: Intel Corporation 82801H (ICH8 Family) USB UHCI #4
00:1b.0 Audio device: Intel Corporation 82801H (ICH8 Family) HD Audio Controller
00:1f.0 ISA bridge: Intel Corporation 82801HB/HR (ICH8/R) LPC Interface Controller
00:1f.2 IDE interface: Intel Corporation 82801H (ICH8 Family) 4 port SATA IDE
        Controller
00:1f.3 SMBus: Intel Corporation 82801H (ICH8 Family) SMBus Controller
01:00.0 VGA compatible controller: nVidia Corporation G70 [GeForce 7600 GS]
02:00.0 Ethernet controller: Marvell Technology Group Ltd. 88E8056 PCI-E Gigabit
        Ethernet Controller (rev 12)
05:03.0 FireWire (IEEE 1394): Texas Instruments TSB43AB22/A IEEE-1394a-2000 Controller
```

Сетевые интерфейсы. Подробная информация о конфигурации интерфейсов WLAN и LAN, а также о работе с модемами сообщается в главе 18.

Bluetooth. Это технология обмена информацией с устройствами по беспроводной связи. Bluetooth не слишком распространена в качестве WLAN и используется преимущественно с сотовыми телефонами, КПК и другими мелкими электронными устройствами. Обычно Linux не составляет труда разобраться с устройствами, применяющими Bluetooth, и каких-либо сложных конфигурационных работ не требуется.

В Gnome при настройке новых устройств по Bluetooth вам пригодятся расположенные на панели программы `bluetooth-applet` и `bluetooth-wizard`. В KDE эти задачи решаются с помощью `kbluetooth` и `kbluetooth-devicemanager`. «За кулисами» при этом действует демон `bluetoothd`, взаимодействующий с `udev` и отвечающий за управление устройствами с Bluetooth. Соответствующие конфигурационные файлы находятся в каталоге `/etc/bluetooth`. Более подробная информация о работе с Bluetooth в Linux приводится на сайте http://www.bluez.org/.

Графика (X). Графические карты используются в Linux системой X Window.

Система горячего подключения

К современным компьютерам можно подключать (а также извлекать из них) жесткие диски, флешки, платы расширения и другие устройства, не выключая при этом компьютер. Linux должна реагировать на изменение ситуации быстро и по возможности автоматически. Эту функцию выполняет *система горячего подключения*, компоненты которой в последние годы все время меняются. Сравнительно недавно из большинства дистрибутивов был удален уровень абстрагирования оборудования (Hardware Abstraction Level, HAL), признанный неэффективным. Кое-где он еще активируется в особых случаях. В настоящее время горячее подключение строится так.

- Ядро — обнаруживает изменения в аппаратном обеспечении, означающие, что пользователь, к примеру, вставил компакт-диск в привод или флешку в компьютер.
- `Udev` — с помощью `udev` ядро создает новые файлы-устройства и запускает нужные программы, чтобы управлять новыми устройствами или отсылать

уведомления в настольную систему. При этом интерпретируются файлы правил из каталогов /lib/udev/rules.d и /etc/udev/rules.d. Советы о составлении или изменении таких правил приводятся здесь: http://www.reactivated.net/writing_udev_rules.html.

- DeviceKit — при управлении некоторыми устройствами и, соответственно, компонентами, помогает так называемый набор для устройств (DeviceKit). Он состоит из библиотек libudev и libgudev, которые, как правило, находятся в одноименных пакетах. Для работы с жесткими дисками (или их секторами) и внешними носителями данных применяются сценарии из пакета udisks (ранее он назывался DeviceKit-disks). Для управления энергопотреблением обычно используются программы из пакета upower (ранее он назывался DeviceKit-power). Более подробная информация содержится на следующих сайтах: http://freedesktop.org/wiki/Software/DeviceKit/; http://www.freedesktop.org/wiki/Software/udisks; http://upower.freedesktop.org/.
- В KDE-4 для обработки уведомлений от HAL и D-Bus используется фреймворк устройств Solid. В Gnome версии 2.22 и выше внешними носителями данных занимается Nautilus вместе с PolicyKit (см. раздел 8.4); конфигурация производится в разделе Правка ▸ Настройки ▸ Медиа.
- D-Bus — это система для обмена уведомлениями между программами. На основе библиотеки libdbus обмен информацией может происходить непосредственно между двумя программами. Если необходимо, чтобы программы обменивались уведомлениями, в качестве центральной точки коммутации применяется служебная программа dbus-daemon.

Аудиосистема (ALSA)

Аббревиатура ALSA означает Advanced Linux Sound Architecture (продвинутая звуковая архитектура Linux). Она присутствует в ядре, начиная с версии 2.6, и отвечает за управление звуковыми картами на самом нижнем уровне. В более ранних версиях ядра для управления звуковыми картами применялась OSS — Open Sound System. ALSA содержит модули snd-pcm-oss, sndseq-oss и snd-mixer-oss, обеспечивающие совместимость с OSS.

Аудиоконтроллер, поддерживаемый ядром, автоматически загружает нужный модуль ALSA. Названия модулей ALSA начинаются с snd. При этом команда lsmod grep snd дает краткий обзор всех активных модулей ALSA. Доступ к различным звуковым функциям осуществляется через файлы, находящиеся в каталоге /proc/asound.

Конфигурация системы ALSA производится в файлах /etc/alsa/*, /etc/asound.conf, а также ~/.asoundrc. Если вы собираетесь использовать аудиосистему как обычно, то менять эти файлы не нужно. Аппаратура должна опознаваться автоматически. Если у вас есть особые требования к аудио (скажем, вы музыкант), то вам нужно, чтобы система различала несколько звуковых карт. Если же у вас есть еще какие-то особые пожелания, обратитесь к следующему сайту и связанному с ним вики-источнику — здесь вы найдете подробную базовую информацию по ALSA и ее конфигурации: http://www.alsa-project.org/.

В случае остановки компьютера или при перезапуске сценарий Init-V (см. главу 16) сохраняет или восстанавливает имеющиеся настройки громкости.

Инструментарий ALSA

Для непосредственного использования ALSA предусмотрено несколько команд (пакет alsa-utils), важнейшие из которых я коротко представлю: `alsactl` сохраняет или загружает все настройки ALSA (громкость звука и т. д.); `alsamixer` изменяет громкость или входной уровень сигнала различных аудиоканалов ALSA; `aplay` воспроизводит аудиофайл; `arecord` записывает аудиофайл.

СОВЕТ

Если колонки молчат, то причина может быть всего лишь в том, что регулятор громкости стоит на нуле. При обычной работе с компьютером важны три канала: ведущая громкость регулирует громкость сигнала в целом; PCM-громкость указывает, какой силы должен быть сигнал аудиофайлов, созданных аудио- и видеопроигрывателями (в данном случае PCM означает «импульсно-кодовая модуляция»); наконец, CD-громкость указывает, как будут вливаться в аудиопоток данные, считываемые непосредственно с CD, если CD-привод и звуковая карта соединены кабелем.

В современных дистрибутивах иногда отсутствуют графические пользовательские интерфейсы, позволяющие по отдельности настраивать сигнал на входах и выходах аудиоустройства. Что делать? Запустите alsamixer в текстовой консоли. Теперь с помощью клавиш управления курсором можно выбирать каналы и настраивать уровни сигнала для них. M (mute) включает или выключает канал.

Аудиобиблиотеки

Многие аудиопрограммы работают с ALSA не напрямую, а обращаются к звуковым библиотекам, звуковым серверам и т. д. Этот промежуточный «слой» между низкоуровневой системой ALSA и самими аудиоприложениями упрощает программирование, приспосабливает эти приложения к работе в сети и гарантирует бесконфликтное выполнение работающих одновременно аудиопрограмм.

Загвоздка лишь в том, что пока не существует единой аудиоархитектуры «выше» системы ALSA. KDE и Gnome работают с системой каждый по-своему. Сложные аудиоприложения, для которых недостаточно имеющихся библиотек, сами внедряют простейшие аудиофункции. Разработать аудиопрограмму, которая бы функционировала самостоятельно, независимо от локального компьютера, исключительно сложно. Специалисты фирмы Adobe сообщили, что разработка Flash-плагина для Linux сильно затянулась во многом из-за того, что имелись серьезные проблемы с аудио.

Далее я кратко представлю некоторые распространенные аудиосистемы.

- **aRts.** Это означает «аналоговый синтезатор реального времени» — основная аудиопрограмма в KDE 2.*n* и 3.*n*. aRts состоит из нескольких модулей, создающих, объединяющих и фильтрующих аудиофайлы. Программы KDE запрашивают aRts с помощью демона `artsd`, который запускается вместе с KDE. Программы, несовместимые с aRts, автоматически получают переадресацию с помощью `artsdsp`. Так или иначе, техническая поддержка aRts уже прекращена, и в версии KDE 4 этой программы уже не будет.
- **EsounD.** Название означает Enlightened Sound Daemon (просветленный звуковой демон) — это была аналогичная aRts программа из Gnome. Совместимые

с EsounD программы посылают аудиофайлы демону esd, запускаемому вместе с Gnome. Судьба EsounD, как и судьба aRts, сложилась незавидно: EsounD все еще устанавливается вместе с Gnome ради совместимости, но большинство аудиоприложений уже перешли к использованию GStreamer.

- **GStreamer.** Данная библиотека — широкий мультимедийный фреймворк (аудио и видео), используемый многими программами Gnome. Благодаря использованию плагинов архитектура программы получилась модульной и рассчитана на значительную расширяемость. В виде плагинов предоставляются и кодеки для обработки различных аудио- и видеоформатов. В отличие от aRts и EsounD, библиотека GStreamer обходится без звукового демона. В данном случае основную работу демона, то есть сведение нескольких звуковых сигналов, теперь выполняет непосредственно ALSA. Более подробная информация представлена на сайте http://www.gstreamer.net/.
- **Phonon.** Мультимедийный фундамент KDE 4 называется Phonon. Эта библиотека содержит унифицированный интерфейс программирования приложений (API) для работы с функциями аудио и видео, который обращается к имеющимся мультимедийным библиотекам (обычно это Xine, но в зависимости от того, какой внутренний интерфейс установлен, это могут быть и GStreamer или VLC). Phonon также применяется библиотекой Qt в качестве мультимедийного интерфейса. Более подробная информация сообщается на сайте Phonon: http://phonon.kde.org/.
- **PulseAudio.** Это звуковой сервер с поддержкой работы в сети, функционально схожий с программой esd и заменяющий ее в современных дистрибутивах. На первый взгляд это незаметно — все должно работать как раньше. Однако «за кулисами» выполняется масса работы: все аудиопотоки могут отдельно управляться программой pavucontrol и присваиваться разным звуковым картам и устройствам вывода. Кроме того, PulseAudio должен автоматически распознавать и активировать дополнительные аудиоустройства, например внешние USB-дисководы. Более подробная информация приводится на следующих сайтах: http://pulseaudio.org/; http://pulseaudio.org/wiki/FAQ.

Наряду с этими аудиосистемами стоит упомянуть несколько программ, которые сами содержат большие аудио- и мультимедийные библиотеки и предоставляют эти библиотеки в пользование другим программам. Наиболее известны RealPlayer или Helix Player на основе Helix DNA, а также видеопроигрыватель Xine на основе xinelib. Неудивительно, что многочисленные несовместимости между различными аудиопрограммами и библиотеками буквально предопределены. Другие причины, по которым аудиосистемы Linux оставляют желать лучшего, обобщены в статье Audio: it's a mess (http://lwn.net/Articles/299211/). Эта статья была написана в сентябре 2008 года, и хотя она уже не новая, но, к сожалению, по-прежнему актуальная.

Для музыкантов или людей, профессионально работающих с аудио, существуют дистрибутивы, специально разработанные для оптимального и безотказного использования аудиопрограмм. В настоящее время наиболее популярна программа Ubuntu Studio: http://ubuntustudio.org/.

11.7. Журналирование

Ядро, различные инструменты администрирования (PAM, APT, `dpkg`) и большинство сетевых служб протоколируют любые мыслимые события в многочисленных файлах, расположенных в каталоге `/var/log`. Эти файлы регистрации событий исключительно удобны при запуске в эксплуатацию новой службы, так как помогают обнаружить ошибки конфигурации. В ходе эксплуатации сервера такие файлы журналирования часто позволяют распознать проблемы с безопасностью.

Программа sysklogd (rsyslogd)

Чтобы каждой программе не пришлось «изобретать велосипед», ядро и некоторые инструменты администрирования обращаются к центральным функциям журналирования, которые объединяются под термином Syslog. В большинстве современных дистрибутивов (Debian, Fedora, openSUSE, Ubuntu) в таком случае применяется программа `rsyslogd`.

Обратите внимание — отнюдь не все сетевые службы пользуются Syslog! В частности, крупные серверные службы, например Apache, CUPS и Samba, используют собственные функции журналирования, интегрированные в программу. Таким образом, эти приложения не поддаются глобальной конфигурации, настраиваемой в `syslog.conf`. Параметры журналирования настраиваются в соответствующих конфигурационных файлах конкретной программы. Если вы хотите составить общее представление о файлах регистрации, управляемых Syslog, выполните следующие команды:

```
root# cd /var/log
root# ls $(find -user syslog).
```

Конфигурация

Конфигурация `rsyslogd` осуществляется в файлах `/etc/rsyslogd.conf` и `/etc/rsyslog.d/*.conf`. Далее я опишу на примерах, как происходит конфигурация в Ubuntu. В этом дистрибутиве большая часть настроек задается в файле `/etc/rsyslog.d/50-default.conf`.

Конфигурационные файлы Syslog содержат правила, состоящие из двух частей. Первая часть (селектор) указывает, что именно должно регистрироваться, а во второй части (действие) описывается, что должно происходить после получения конкретного уведомления. Правила можно разделять на несколько строк с помощью символа \. Возможны ситуации, в которых одному уведомлению соответствует несколько правил. В этом случае уведомление выдается и регистрируется несколько раз.

Каждый **селектор** состоит из двух частей, разделенных точкой: *служба.уровень_приоритета*. Можно указывать сразу несколько селекторов, разделяя их точками с запятой. Кроме того, можно объединять *в одном селекторе* несколько служб, разделяя их запятыми. Все программы Linux, использующие Syslog, должны присваивать своим сообщениям службу и приоритет.

Syslog известны следующие службы: uth, authpriv, cron, daemon, ftp, kern, lpr, mail, news, syslog, user, uucp, а также службы от local0 до local7. Символ * объединяет все службы.

Кроме того, Syslog известны следующие степени приоритета (по увеличению важности): debug, info, notice, warning = warn, error = err, crit, alert и emerg = panic. Ключевые слова warn, error и panic считаются устаревшими — используйте вместо них warning, err и emerg. Символ * объединяет все степени приоритета. Ключевое слово none применяется с уведомлениями, которым не присвоен никакой приоритет.

При указании степени приоритета учитываются также все степени, находящиеся в иерархии выше. Иными словами, селектор mail.err охватывает и все сообщения почтовой системы уровней crit, alert и emerg. Если вам требуются уведомления только с определенной степенью приоритета, поставьте перед этой степенью символ = (то есть, например, mail.=err).

В качестве **действия** чаще всего указывается имя одного из файлов регистрации. Обычно файлы регистрации синхронизируются после каждого вывода. Если перед именем файла стоит символ -, Syslog не производит синхронизацию. Такой метод работы более эффективен, но при аварийном завершении работы системы будут потеряны сообщения, еще не сохраненные физически.

Syslog также может переадресовывать уведомления к файлам FIFO (первым пришел — первым обслужен) или к программным каналам. В таком случае перед именем файла ставится символ |. Файл /dev/xconsole, показанный в следующем листинге, — это особый файл типа FIFO, предназначенный для передачи уведомлений графической системе X.

Символ * означает, что уведомление переадресовывается на все консоли или всем пользователям, вошедшим в систему через SSH. Поскольку это очень неудобно, по умолчанию он применяется только с критически важными сообщениями.

Более подробная информация по синтаксису syslog.conf содержится на одноименной странице справки man. В следующих строках приводится немного сокращенная стандартная конфигурация Syslog в Ubuntu, отформатированная для удобства:

```
# Файл /etc/rsyslog.d/50-default.conf bei Ubuntu
# Селектор                  Действие
auth,authpriv.*             /var/log/auth.log
*.*;auth,authpriv.none      -/var/log/syslog
kern.*                      -/var/log/kern.log
mail.*                      -/var/log/mail.log
mail.err                    /var/log/mail.err
*.emerg                     :omusrmsg:*
daemon.*;mail.*;\
       news.err;\
       *.=debug;*.=info;\
       *.=notice;*.=warn    |/dev/xconsole
```

Рассмотрим, что означает этот код.

- /var/log/auth содержит аутентификационные сообщения со всеми степенями приоритета. Сюда относятся как удачные, так и неудачные попытки входа в систему (в том числе по SSH), сообщения PAM, команды sudo и т. д. Поскольку

единственным файлом регистрации является auth, он сразу же синхронизируется при каждом сообщении.

- /var/log/syslog включает в себя все сообщения, зарегистрированные Syslog (в том числе аутентификационные, которым не присвоен приоритет). Такой всеобъемлющий подход и хорош, и плох одновременно. С одной стороны, вы сможете получить из единственного файла любую информацию, которая вам понадобится. С другой — в такой «куче» исключительно сложно найти нужные записи.
- /var/log/kern.log содержит все сообщения ядра.
- Уведомления почтовой системы (например, Postfix) подразделяются на несколько файлов. В mail.log сохраняются *все* уведомления, в mail.err — только сообщения об ошибках.
- /var/log/debug содержит уведомления, связанные с отладкой системы, а также сообщения для служб auth, authpriv, news и mail без указания приоритета.
- Критические системные сообщения (например, о близящемся «обвале» системы или об ошибке ядра) переадресовываются с помощью :omusrmsg:* всем пользователям (то есть во все окна терминалов, на все консоли). Omusrmsg — это модуль rsyslog, отвечающий за отправку уведомлений пользователям.
- Различные предупреждения и сообщения об ошибках передаются в систему X. Чтобы отслеживать такие сообщения в X, запустите программу xconsole. Она выглядит как маленькое окно терминала, но вводить в нее информацию нельзя.

Журналирование ядра

Сообщения ядра записываются в кольцевой буфер размером 16 Кбайт, расположенный в оперативной памяти. Когда этот буфер заполнен, старые сообщения удаляются, освобождая место для новых. Содержимое этого буфера можно просмотреть с помощью команды dmesg. Если указать при этом параметр -c, кольцевой буфер будет автоматически опустошаться.

Кроме того, все уведомления ядра записываются в виртуальный файл /proc/kmsg. Он служит для переадресации сообщений ядра в Syslog.

Часто сообщениям ядра (dmesg) предшествует указание времени в форме [nnn.nnnnnn]. Число перед точкой указывает количество секунд, истекших с запуска системы, а следующие шесть знаков конкретизирует время вплоть до миллионной доли секунды. При сохранении уведомлений ядра в файле регистрации такой показатель времени обычно дополняется указанием абсолютного времени.

Сообщения Init

Сообщения системы Init (см. главу 16) регистрируются лишь в некоторых дистрибутивах. К таким приятным исключениям относится Fedora — здесь регистрация происходит в файле /var/log/boot.log.

Программа logrotate

Как правило, файлы регистрации медленно, но верно разрастаются, занимая все больше места. Если хотите контролировать размер файлов регистрации, пользуйтесь

программой logrotate. Она ежедневно вызывается сценарием Cron /etc/cron.daily/logrotate. Затем сценарий обрабатывает все файлы регистрации, описанные в конфигурационных файлах каталога /etc/logrotate.d. Подробности обработки файлов регистрации программой logrotate зависят от конкретной конфигурации программы. Однако в целом этот процесс протекает одинаково.

1. Программа logrotate переименовывает имеющийся архив файлов регистрации. Из *имя*.4.gz получается *имя*.5.gz, из *имя*.3.gz — *имя*.2.gz и т. д. Этот процесс называется *ротацией*. При этом пакету присваивается собственное имя.
2. Если количество пакетов с файлами регистрации превышает максимальную заданную величину, то более старые архивные файлы удаляются.
3. Программа logrotate переименовывает актуальный файл регистрации. Этот файл теперь называется *имя*.0.
4. Программа создает новый, пустой файл регистрации имя.
5. При работе с большинством серверных служб logrotate с помощью команды /etc/init.d/*имя* reload требует от демона заново считать конфигурацию. Демон понимает, что появился новый пустой файл регистрации, и работает именно с ним.
6. Программа logrotate архивирует *имя*.0 или *имя*.1 (параметр delaycompress). Указанный параметр позволяет избежать конфликтов с демоном, который, возможно, еще записывает информацию в *имя*.0, и командой архивации (обычно это gz).

Конфигурация. В /etc/logrotate.conf содержатся некоторые установки, действующие для logrotate по умолчанию. Эти настройки действительны, если программно-специфичные файлы конфигурации не содержат иных данных.

В /etc/logrotate.d содержатся стандартные настройки для различных программ, создающих файлы регистрации. Эти файлы относятся не к пакету Logrotate, а к пакетам соответствующих программ. Так, например, пакет samba расположен в каталоге /etc/logrotate.d/samba. Это позволяет гарантировать, что файлы устанавливаемой в настоящий момент программной версии подходят, и logrotate информирует каждую службу о переименовании файлов регистрации либо запускает ее заново.

В следующих строках показан пример конфигурационного файла Apache в Ubuntu: программа logrotate обрабатывает файлы регистрации раз в неделю. Файлы регистрации переименовываются и архивируются. Размер архива ограничен 52 файлами, то есть при необходимости вы можете просмотреть файлы регистрации за целый год.

```
# Файл /etc/logrotate.d/apache
/var/log/apache2/*.log {
  weekly
  missingok
  rotate 52
  compress
  delaycompress
  notifempty
```

```
  create 640 root adm
  sharedscripts
  postrotate
    if [ -f "`. /etc/apache2/envvars ; \
      echo ${APACHE_PID_FILE:-/var/run/apache2.pid}`" ]; then
      /etc/init.d/apache2 reload > /dev/null
    fi
  endscript
}
```

Пока Apache работает, команда `reload` сообщает ему, что появились новые файлы регистрации.

Logwatch

Самостоятельное изучение файлов регистрации на первых порах или при поиске какой-либо ошибки может быть очень интересным. Но вскоре вы, как и любой другой администратор сервера, утратите тягу к изучению файлов регистрации и будете относиться к такой информации все менее и менее внимательно! Здесь вам помогут специальные инструменты для автоматической интерпретации журналирования. В качестве примера я расскажу о программе `logwatch`, которая присутствует в большинстве дистрибутивов в виде отдельного пакета.

После установки программа `logwatch` выполняется один раз в день посредством `/etc/cron.daily/*logwatch`. Она интерпретирует записи, занесенные в файлы регистрации на протяжении последних 24 часов, обобщает важную информацию в электронном сообщении и отсылает это письмо администратору (root).

ПРИМЕЧАНИЕ

Для работы с logwatch на компьютере нужно установить ПО почтового сервера, который будет отправлять обобщенные данные о журналировании! Если при инсталляции logwatch на компьютере имеется только Postfix и вы хотите заниматься лишь обработкой локальной почты (иными словами, отказываетесь от настройки почтового сервера), установите флажок Only Local (Только на локальной машине). По умолчанию электронные сообщения для администратора переадресуются только тому пользователю, который указан в качестве стандартного в /etc/aliases.

Обобщенная информация о журналировании обычно занимает несколько страниц — собственно, это слишком много, чтобы ежедневно прочитывать ее самостоятельно. Объем текста зависит от того, сколько серверных служб установлено на компьютере и сколько событий попало в журнал регистрации за последний день. Далее приводится электронное сообщение `logwatch`, сокращенное ради экономии места:

```
Logwatch 7.4.0 (03/01/11)
Processing Initiated: Tue Jul 10 06:25:02 2012
Date Range Processed: yesterday
                  ( 2012-Jul-09 )
                  Period is day.
Detail Level of Output: 0
Type of Output/Format: mail / text
Logfiles for Host: xxxx
```

```
--------------------- Denyhosts Begin ------------------------

new denied hosts:
   60.12.251.5
   222.45.235.70
--------------------- pam_unix Begin ------------------------
sshd:
   Authentication Failures:
      root (222.45.235.70): 7 Time(s)
      root (60.12.251.5): 3 Time(s)
      unknown (60.12.251.5): 1 Time(s)
   Invalid Users:
      Unknown Account: 1 Time(s)

--------------------- Postfix Begin ------------------------

   3.347K   Bytes accepted                    3,427
   2.343K   Bytes sent via SMTP               2,399
   4.910K   Bytes delivered                   5,028

--------------------- SSHD Begin ------------------------

Failed logins from:
   60.12.251.5: 3 times
   222.45.235.70: 7 times
Illegal users from:
   undef: 1 time
   60.12.251.5: 1 time
Received disconnect:
   11: Bye Bye [preauth] : 11 Time(s)
Refused incoming connections:
   222.45.235.70 (222.45.235.70): 1 Time(s)
   60.12.251.5 (60.12.251.5): 1 Time(s)

--------------------- Disk Space Begin ------------------------

Filesystem              Size  Used  Avail Use%  Mounted on
/dev/mapper/vg0-root    20G   1.2G  18G   6%    /
udev                    7.8G  4.0K  7.8G  1%    /dev
/dev/mapper/vg0-var     20G   723M  19G   4%    /var
/dev/mapper/vg0-images  20G   6.6G  13G   35%   /var/lib/libvirt/images
/dev/mapper/vg0-backup  100G  12G   84G   13%   /backup
/dev/md0                1016M 75M   890M  8%    /boot
```

Конфигурация. Может возникнуть вопрос: а почему `logwatch` не требует никакой конфигурации, и начинает работать сразу, как по волшебству? Причина проста: вместе с `logwatch` устанавливается и ее стандартная конфигурация, которая находится в каталоге `/usr/share/logwatch/default.conf`. Расположенный здесь файл `logwatch.conf` содержит некоторые глобальные настройки. В частности, тут можно указать, кому будет направляться обобщенная информация (`MailTo = root`), за какой

период будут обрабатываться файлы регистрации (`Range = yesterday`), насколько детальным должно быть описание (`Detail = Low`) и т. д.

Кроме того, в файлах `default.conf/services/*.conf` содержатся настройки конфигурации для разнообразных серверных служб, в частности, для Apache, Postfix, ClamAV, Dovecot, Sendmail, SSH и др. Разумеется, эти конфигурационные файлы используются лишь при условии, что на компьютере установлена та или иная служба. Кроме того, для получения релевантных результатов необходимо не изменять стандартного расположения файлов журналирования.

В каталоге `/usr/share/logwatch/dist.conf` содержатся специфичные для конкретного дистрибутива изменения, отличающие действующую конфигурацию от стандартной. Сделанные здесь изменения имеют приоритет перед стандартной конфигурацией.

Чтобы самостоятельно изменить конфигурацию, скопируйте интересующий вас файл в соответствующий каталог в `/etc/watchlog` и измените там этот файл. В файле могут содержаться только изменения, отличающие его от оригинала — и ничего больше. В качестве эксперимента можете запустить `logwatch` вручную:

```
root# logwatch --mailto name@host
```

12 Управление программами и пакетами

В Windows программы обычно устанавливаются с помощью исполняемого файла `setup.exe`. В Linux применяется совершенно другой подход: система управления пакетами руководит базой данных, в которой содержится информация обо всех уже инсталлированных программных пакетах. Новые программы устанавливаются с помощью команд системы управления пакетами.

Такая концепция имеет много преимуществ: можно учитывать взаимозависимости и конфликты между программными пакетами. Если, к примеру, для работы программы *A* требуется библиотека *B*, то система управления пакетами разрешит установить программу *a* только после того, как вы установите библиотеку *B*. В любое время можно определить, к какому пакету относится тот или иной файл, находится ли еще этот файл в исходном состоянии и т. д.

Форматы пакетов. На рынке Linux доминируют две системы управления пакетами: Red Hat, Fedora, Mandriva, SUSE. Другие многочисленные дистрибутивы используют формат пакетов RPM, разработанный Red Hat. Debian и другие дистрибутивы, созданные на его основе, применяют, напротив, формат DEB. Команды для инсталляции, деинсталляции и обновления этих пакетов (`rpm`, `dpkg` и т. д.) в любом случае достаточно примитивны. Например, они позволяют определить недостающие зависимости, но не позволяют решить связанные с этим проблемы.

Системы управления пакетами. На основе `rpm` или `dpkg` были разработаны другие программы с различными дополнительными функциями. К таким функциям относится, в частности, автоматическая установка зависимых пакетов, возможность обновлений для всей системы и учет источников пакетов, находящихся на CD/DVD или в Интернете. Кроме того, система управления пакетами позволяет создать унифицированную систему обновления для *всех* установленных программ. К подобным системам управления пакетами относятся, в частности, Yum и ZYpp — в формате RPM, а также APT и Aptitude в формате DEB . Графические пользовательские пакеты также имеются в изобилии. Популярны программы YumExtender, PackageKit и Synaptic.

Инструменты, специфичные для отдельных дистрибутивов. В качестве дополнения к этим стандартным программам в некоторых дистрибутивах имеются собственные программы для управления пакетами и выполнения обновлений:

- Debian, Ubuntu — `update-manager`;
- Fedora версии 9 и выше, RHEL 6 — `packagekit`;
- Fedora версии 8 и ниже, RHEL 5 — `pirut` и `pup`;
- SUSE — YaST-модули из группы Программное обеспечение;
- Ubuntu — `gnome-app-install`, `gnome-language-selector`.

ВНИМАНИЕ

Пакеты каждого отдельного дистрибутива Linux согласованы друг с другом. Это означает, что они используют одинаковые библиотеки, собираются одним и тем же компилятором и т. д. Поэтому рекомендую всем начинающим работу с Linux устанавливать только те пакеты, которые предназначены специально для вашего дистрибутива. Так, например, может не получиться инсталлировать пакет для Red Hat в SUSE (или наоборот). Для устранения этих проблем обычно требуются глубокие специальные знания.

Безопасность. В большинстве систем управления пакетами предпринимаются серьезные меры обеспечения безопасности. В частности, пакеты сами по себе и описание источников пакетов снабжаются криптографическими подписями, исключающими манипуляции с ними. Поэтому при создании новых источников пакетов зачастую требуется установить новый ключ. Подробный анализ аспектов безопасности систем управления пакетами различных дистрибутивов делается в следующей статье: http://lwn.net/Articles/326817/.

Недостатки управления пакетами. Управление пакетами в распространенных дистрибутивах работает хорошо, но характеризуется некоторыми недостатками.

- Практически ежедневные обновления, обычные для многих дистрибутивов, непривычны для пользователей, ранее работавших с Windows или Mac OS X и привыкших, что обновления выходят гораздо реже.
- Загрузка обновлений — довольно ресурсозатратный процесс, и для работы в режиме постоянных обновлений требуется быстрое соединение с Интернетом. Вполне возможно, что вам ежемесячно придется скачивать многие сотни мегабайт обновлений.
- Многие пользователи Linux предпочитают подолгу применять один и тот же дистрибутив, но обновляют при этом лишь немногие программы (обычно это настольные приложения, такие, как LibreOffice или Gimp). Именно такие функции работают в распространенных дистрибутивах хуже некуда. Правда, время от времени предлагаются обновления для системы безопасности, но автоматический переход на новую версию невозможен. Если вы хотите перейти на новую версию определенной программы, то придется произвести установку вручную, обратиться к неофициальным источникам пакетов или переустановить весь дистрибутив. Это касается, в частности, установки новой версии LibreOffice. В Windows и Mac OS X процесс обновления, наоборот, упрощен до предела.

  Такая ситуация объясняется причинами технического характера. Дело в том, что большинство программ Linux использует бесчисленные библиотеки. Для обновления версии LibreOffice, как правило, требуется обновить некоторые библиотеки. Из-за этого могут возникнуть проблемы с другими программами, которые также обращались к старой версии измененной библиотеки.

Возможный выход заключается в том, чтобы сразу интегрировать в важные программы относящиеся к ним библиотеки. (В Google Chrome такой подход применялся изначально, поддержка пакетов Firefox и Thunderbird в современных дистрибутивах также организована таким образом.) Но и этот подход не лишен недостатков: из-за избыточности, избавиться от которой пока невозможно, обновления требуют все больше места на диске, объем скачиваемого материала возрастает при каждой новой загрузке. Кроме того, увеличивается объем оперативной памяти, необходимый для одновременного выполнения нескольких программ. Но самые большие сложности возникают с технической поддержкой: каждая программа теперь управляет своими библиотеками самостоятельно. Если в какой-то библиотеке возникает проблема с безопасностью, эту проблему уже нельзя устранить централизованно. Приходится обновлять все программы, использующие данную библиотеку.

Централизованное администрирование нескольких компьютеров. Если вы отвечаете за 50, 100 или 1000 компьютеров с Linux, то администрирование и управление пакетами станет для вас мукой, несмотря на все разнообразие инструментов, описанных в этой главе. Вам понадобится инструмент, который позволил бы проводить обновления на всех компьютерах сети или некоторых предварительно выбранных машинах, разрешил установить новую программу или изменить конфигурацию. В зависимости от дистрибутива для этих целей применяются такие программы, как Red Hat Network, ZENworks (Novell/SUSE) m23 (Debian) или Orchestra или Landscape (обе — Ubuntu):

- http://www.redhat.com/rhn/;
- http://www.novell.com/de-de/products/zenworks/configurationmanagement/;
- http://m23.sourceforge.net/;
- https://launchpad.net/orchestra;
- http://www.canonical.com/enterprise-services/ubuntu-advantage/landscape.

Управление пакетами с помощью tar. Команда `tar` позволяет собрать множество файлов в одном архиве либо распаковать такой архив. На заре Linux, когда еще не было пакетов в форматах RPM и DEB, в большинстве дистрибутивов вместо пакетов использовались TAR-архивы. Сегодня осталось совсем мало дистрибутивов, в которых применяются TAR-архивы, а не пакеты (например, Slackware).

И все же TAR-архивы еще играют важнейшую роль в повседневном труде опытных пользователей Linux. Многие разработчики программ, не желающие создавать пакеты RPM или DEB, просто предоставляют вместо этого обычные TAR-архивы со всеми необходимыми файлами. Установить такие файлы не сложнее, чем обычный пакет (`tar czf` *`имя`*`.tgz`), но при этом программа ускользает от системы управления пакетами конкретного дистрибутива, ее сложно обновлять, нельзя деинсталлировать, и к тому же она может вызывать конфликты. Рекомендую устанавливать TAR-пакеты лишь в тех случаях, когда вы знаете, что делаете, и когда желаемая программа недоступна ни в какой другой форме.

Другие форматы пакетов. Кроме рассматриваемых в этой книге пакетов RPM и DEB, существуют некоторые другие форматы, которые, правда, пока не приобрели существенного значения. К важнейшим целям, которых разработчики пыта-

ются достичь с помощью новых пакетов, относится обеспечение более тесной связи с дистрибутивом, а также возможность установки пакетов или программ в локальные пользовательские каталоги, не требующая при этом наличия прав администратора.

Более подробную информацию можно найти на следующих сайтах http://labix.org/smart; http://0install.net/.

12.1. Управление пакетами RPM

Команда `rpm` позволяет установить RPM-пакеты и управлять ими. Она помогает, в частности:

- в рамках установки автоматически вносить изменения в уже имеющиеся файлы (например, в сценарные файлы);
- заменить текущую версию программы более новой (причем измененные файлы автоматически обновляются);
- удалить все файлы определенной программы;
- убедиться, что перед установкой программы выполнены все необходимые условия (имеются все требуемые библиотеки необходимых версий);
- проверить, был ли изменен файл с момента установки пакета;
- определить, к какому пакету относится определенный файл.

Информация, необходимая при управлении, содержится в каждом RPM-пакете. При установке такая информация заносится в базу данных (файлы каталога `/var/lib/rpm`).

Происхождение и обслуживание. Формат RPM первоначально разрабатывался для Red Hat. После того как в течение многих лет код поддерживался без энтузиазма, эта разработка раскололась на два самостоятельных проекта. Используемая в Fedora, Red Hat, SUSE и т. д. версия RPM 4.*n* теперь поддерживается и развивается под руководством Fedora и в кооперации с другими дистрибутивами в рамках проекта rpm.org (http://rpm.org/). Параллельно другая группа разработчиков занимается RPM 5.*n* (http://rpm5.org/). Однако эта версия в настоящее время применяется лишь в немногих относительно небольших дистрибутивах. Самый известный представитель группы RPM-5 — дистрибутив Mandriva (http://rpm.org/ (RPM 4.n); http://rpm5.org/ (RPM 5.n)).

Основы

Большинство RPM-пакетов предоставляются в двух вариантах: в виде двоичного пакета и в виде пакета с исходным кодом. Двоичный пакет содержит файлы, необходимые для исполнения программы. Пакет с исходным кодом интересен для разработчиков. Он хранит код, который необходим для создания двоичного пакета.

Уже в названии пакета содержится немало информации: например, `abc-2.0.7-1.i686.rpm` означает пакет `abc` версии 2.0.7 выпуска 1 (если при составлении пакета

возникла ошибка, была добавлена дополнительная онлайн-документация либо были внесены другие изменения, то номер выпуска больше 1 на число, равное номеру конкретной версии, то есть номер версии касается самой программы, а номер выпуска — сборки rpm).

Обозначение i686 указывает на то, что пакет содержит двоичные файлы для процессоров, совместимых с Pentium-II (разумеется, существуют версии Linux и для других процессоров). Если пакет abc содержит сценарные или текстовые файлы, не зависящие от архитектуры процессора, то вместо обозначения процессора используется сокращение noarch. Если пакет содержит исходный код, то здесь применяется сокращение src.

Кратко объясню различные сокращения x86:

- i386 = 386 и совместимые с ним процессоры;
- i486 = 486 и совместимые с ним процессоры;
- i586 = Pentium и совместимые с ним процессоры;
- i686 = Pentium II и совместимые с ним процессоры;
- x86_64 = AMD64 и совместимые с ним 64-битные процессоры (включая Intel).

Пакет i386 будет работать со всеми вышеуказанными процессорами. Напротив, пакет i686 использует различные расширения Pentium II, поэтому код может выполняться немного быстрее. В любом случае программа не будет работать с другими процессорами.

Метаданные

Кроме файлов, предназначенных для установки, файл пакета содержит подробную информацию по администрированию: краткое описание пакета, информацию о номерах версий, о месте в групповой иерархии, информацию о взаимозависимости с другими пакетами и т. д. Взаимозависимости возникают, если для работы пакета требуется определенный язык программирования (например, Perl) или определенная библиотека. В таком случае сначала потребуется установить необходимые пакеты.

Программа rpm управляет базой данных, содержащей информацию обо всех установленных двоичных пакетах. Эта база данных сохраняется в виде отдельных файлов в каталоге /var/lib/rpm. В базе данных содержится информация только относительно двоичных пакетов; если у вас установлены пакеты с двоичным кодом, они не учитываются в базе данных.

Файлы базы данных RPM ни в коем случае нельзя изменять непосредственно! Чтобы база данных RPM содержательно совпадала с действительной конфигурацией установки, пакеты необходимо удалять не простым удалением файлов, а с помощью процедуры деинсталляции (rpm -e).

Пакеты Delta-RPM

Чтобы обновить пакет RPM, следует целиком загрузить новый пакет. И именно при проведении обновлений для системы безопасности, когда зачастую требуется внести незначительнейшие изменения всего в несколько файлов, такой метод неэффективен. Поэтому в SUSE были разработаны так называемые *пакеты Delta-RPM*,

в которых хранятся изменения, видимые лишь в сравнении с определенной версией пакета. В Mandriva и Fedora эта идея также прижилась и теперь используется при обновлениях.

В принципе, применять пакеты Delta-RPM очень просто: сначала команда `applydeltarpm` создает из пакета Delta-RPM и оригинального пакета (либо относящихся к нему установленных файлов) новый, обновленный RPM-пакет. Он устанавливается как обычно (`rpm -U`). Команда `applydeltarpm` входит в состав пакета `deltarpm`.

Однако у Delta-RPM есть и недостатки: `applydeltarpm` потребляет очень много ресурсов процессора и для ее работы необходимо, чтобы была установлена совершенно конкретная версия пакета. Если это не так либо если файлы этой версии после установки были изменены, потребуется обновить оригинальный файл RPM.

Проблемы

Система RPM — совершенно великолепная вещь, но и она имеет свои слабые стороны. Перечислю самые распространенные проблемы.

- Хотя в пакетах RPM и сохраняются межпакетные взаимосвязи, команда `rpm` не в состоянии автоматически их отменять. Для этого используются, в первую очередь, такие программы, как Yum.
- Управление взаимозависимостями пакетов нарушается, если смешиваются пакеты из разных дистрибутивов, если вы устанавливаете отдельные программы с применением `tar` или если вы сами компилируете программы. Причина все та же: либо информация о том, какие программы установлены на компьютере, отсутствует в принципе, либо данные не стыкуются (так как в каждом дистрибутиве действуют собственные принципы формулирования межпакетных взаимосвязей).
- Новые версии RPM хотя и обладают обратной совместимостью, но не совместимы «снизу вверх». Это означает, что, например, команда `rpm` версии 4.4 не может работать с файлами RPM, созданными с помощью версии 4.7. Что делать? Установите новейшую версию `rpm`.
- Дисковое пространство, необходимое для базы данных RPM (то есть для файлов каталога `/var/lib/rpm`), достаточно велико. При установке программ общим объемом 5 Гбайт файлы базы данных RPM займут около 80 Мбайт.
- Иногда база данных RPM содержит противоречивую информацию. В результате становится невозможно пользоваться командой `rpm` либо выдаются сообщения об ошибках вроде «невозможно открыть пакеты базы данных». В таких случаях обычно помогает команда `rm -f/var/lib/rpm/__db*`, а потом — `rpm --rebuilddb`. Тогда база данных RPM создается заново. Правда, на это требуется некоторое время.

Примеры

В следующих примерах показаны типичные ситуации, в которых используется команда `rpm`.

- Установка программы abc:

```
root# rpm -i abc-2.0.7-1.i686.rpm
```

- Загрузка файла ключа GPG с http://myserver.com и приведение его в форму файла ключа для RPM:

```
root# rpm --import http://myserver.com/RPM-myserver-GPG-KEY
```

- Обновление программы abc, причем файл пакета скачивается прямо с указанного сервера:

```
root# rpm -U http://myserver.com/mypath/abc-2.1.0-2.i686.rpm
```

- Вывод списка всех файлов, входящих в abc, которые были изменены с момента установки:

```
root# rpm -V abc
```

- Удаление программы abc:

```
root# rpm -e abc
```

- Составление списка всех установленных пакетов:

```
root# rpm -qa
```

- Составление списка всех установленных пакетов, причем пакеты сортируются по дате установки (чем позже был установлен пакет, тем выше в списке он расположен):

```
root# rpm -qa --last
```

- Составление списка всех установленных пакетов, в названии которых содержится последовательность символов mysql (в любом регистре):

```
root# rpm -qa | grep -i mysql
```

- Выдача информации о пакете perl (если он установлен):

```
root# rpm -qi perl
```

- Перечисление всех файлов из пакета perl:

```
root# rpm -ql perl
```

- Перечисление всех файлов документации из пакета perl:

```
root# rpm -qd perl
```

- Перечисление всех конфигурационных файлов пакета cups:

```
root# rpm -qc cups
```

- Выдача информации об еще не установленном пакете. Для этого на локальном компьютере должен быть доступен RPM-файл:

```
root# rpm -qip abc-2.0.7-1.i686.rpm
```

- Указание, из какого пакета взят файл /usr/lib/libz.so (результат: kdelibs):

```
root# rpm -qf rpm -qf /usr/lib/libkdnssd.so
```

- Указание, какие атрибуты предоставляет пакет `php-mysql`. Результат в Fedora 11: `ysql.so`, `mysqli.so`, `pdo_mysql.so`, `php-mysqli`, `php_database` и `php-mysql=5.2.9-2`. Атрибуты применяются для отмены межпакетных взаимосвязей. Обычно названия атрибутов соответствуют названиям программ или библиотек, предоставляющих пакет. Так или иначе, от администратора пакетов любого дистрибутива зависит, каким образом будут определяться атрибуты. Атрибут также может содержать номер версии.

  ```
  root# rpm -q --provides php-mysql
  ```

- Указание, какие условия должны выполняться для установки пакета `php-mysql`. В Fedora конечный список атрибутов получается очень длинным, поэтому здесь я приведу только выдержки из него: `libc.so.6`, `libm.so.6`, `libmysqlclient.so.16`, `php-common=5.2.9-2` и `php-pdo`.

  ```
  root# rpm -q --requires php-mysql
  ```

- Команды `rpm -q --provides` или `rpm -q -provides` также можно комбинировать с параметром `-p`, чтобы узнавать списки атрибутов для еще не установленных пакетов (в предыдущем примере — `alien`):

  ```
  root# rpm -q --requires -p alien-8.56-2.i586.rpm
  ```

- Указание, какой из уже установленных пакетов предоставляет атрибут `mysqli.so` (результат: пакет `php-mysql`):

  ```
  root# rpm -q --whatprovides mysqli.so
  ```

- Выдача практически бесконечного списка всех установленных пакетов, зависящих от атрибута `libpthread.so.0` (кстати, одноименная библиотека предоставляется пакетом `glibc`):

  ```
  root# rpm -q --whatrequires libpthread.so.0
  ```

Проблемы 32/64 бит

В 64-битных дистрибутивах может случиться так, что команда `rpm -qi name` выдаст не название конкретного пакета, а информацию по *двум* пакетам. Это не ошибка — по-видимому, речь идет о двух одноименных пакетах с файлами для 32- и 64-битного вариантов программы или библиотеки.

SUSE старается обходиться без применения одноименных пакетов с разным содержимым, помечая 32-битные варианты пакетов окончанием `32bit`. В этом дистрибутиве команда `rpm -qa grep 32bit` возвратит удивительно длинный список всех 32-битных пакетов — такие пакеты нужны по причинам, связанным с совместимостью.

Взаимозависимости пакетов

Вероятно, вы впервые столкнетесь с взаимозависимостями пакетов, когда при попытке установить пакет получите сообщение об ошибке, которое будет гласить: failed dependencies: attributname is needed by paketname (неверные зависимости: для работы *название_атрибута* требуется *название_пакета*).

Чтобы решить такие проблемы, при установке лучше использовать не RPM, а Yum, Zypper или другой инструмент управления пакетами, предусмотренный в вашем дистрибутиве. Если это невозможно, придется поискать пакет, предоставляющий *название_атрибута*. Для этого можно воспользоваться поисковиком http://rpmfind.net, позволяющим искать файлы и атрибуты, которые содержатся в пакете RPM.

12.2. Yum

Программа Yum облегчает управление пакетами. Она основана на RPM и предоставляет множество дополнительных функций.

- В качестве источников данных (репозиториев) служат архивы Yum, расположенные в Интернете. Yum-архив — это собрание пакетов RPM, дополнительно снабженных метаданными (каталог repodata). В метаданных находится информация о содержимом и зависимостях всех пакетов. Благодаря разделению пакетов и метаданных обработка информации ускоряется (так как не нужно считывать все пакеты, чтобы извлечь данные). Yum способна автоматически переходить между несколькими зеркалами, на которых расположен источник пакетов.
- Yum автоматически отменяет все взаимозависимости пакетов, загружает все необходимые пакеты и устанавливает их. При этом учитываются все источники пакетов. Если вы, например, устанавливаете пакет из репозитория *A*, Yum может предварительно загрузить определенные связанные с ним пакеты из источников *B* и *C*.
- Программа Yum может обновить все установленные пакеты одной-единственной командой. Для этого она проверяет, имеются ли в доступных источниках обновленные версии установленных пакетов. Если это так, то необходимые пакеты скачиваются и устанавливаются. Разумеется, при этом снимаются и все взаимозависимости пакетов.

Yum была разработана на языке программирования Python. Программа по умолчанию применяется в том числе в Fedora и в Red Hat. В этом разделе будет рассмотрено применение Yum в Fedora 11.

Более подробная информация по Yum дается по адресу http://www.linux.duke.edu/projects/yum/.

Конфликты с блокировкой. Нельзя одновременно запускать несколько образцов Yum. Если уже работает программа или команда Yum, то при повторном запуске выводится сообщение об ошибке another copy is running (работает другой экземпляр).

Суть проблемы заключается в работе системы автоматических обновлений Yum. Этот процесс несложно обнаружить: файл `/var/run/yum.pid` содержит ID-номер программы, вызывающей конфликт. С помощью команды `ps grep id` можно узнать имя программы. При необходимости можно временно остановить систему обновлений с помощью команды `/etc/init.d/yum-updatesd stop`, а затем вновь запустить ее командой `start`.

Конфигурация

Основная конфигурация Yum выполняется в файле /etc/yum.conf. В следующих строках показаны выдержки из конфигурации в системе Fedora 12:

```
# Файл /etc/yum.conf
[main]
cachedir=/var/cache/yum/$basearch/$releasever
keepcache=0
debuglevel=2
logfile=/var/log/yum.log
exactarch=1
obsoletes=1
gpgcheck=1
plugins=1
installonly_limit=3
color=never
```

Кратко опишу важнейшие настройки. Благодаря keepcache=0 скачанные пакеты не архивируются после установки. Как правило, эта установка полезна, так как дисковое пространство, необходимое для пакетов, со временем сильно вырастет, и, скорее всего, не будет причин устанавливать пакеты повторно. В любом случае Yum может обнаружить в ходе установки проблему и прервать установку. Если есть возможность устранить проблему и повторно произвести обновления, все пакеты потребуется скачать заново. Чтобы избежать этого, укажите keepcache=1. Чтобы специально удалить пакеты, находящиеся в /var/cache/yum, выполните команду yum clean packages.

При наличии exaktarch=1 Yum учитывает только такие обновления, архитектура которых соответствует архитектуре уже установленного пакета. Иначе говоря, заменить пакеты i386 новыми пакетами x86_64 не получится.

При использовании gpgcheck=1 Yum проверяет ключ аутентификации пакета. Переменную gpgcheck также можно настроить иначе, чем указано в yum.conf — отдельно для каждого репозитория. Переменная plugins определяет, будет ли Yum учитывать плагины.

Некоторые пакеты Yum должна установить, но не должна обновлять. К ним относятся, в частности, пакеты ядра: при обновлении ядра дополнительно устанавливается новый пакет ядра, не затрагивая при этом старый пакет. Переменная installonlypkgs позволяет указывать названия таких пакетов. По умолчанию она имеет настройки kernel, kernel-smp, kernel-bigmem, kernel-enterprise, kernel-debug, kernel-unsupported. Наконец, переменная installonly_limit, содержащаяся в yum.conf, указывает, сколько версий подобных пакетов может быть установлено параллельно. При стандартной настройке одновременно остаются установлены только три новейшие версии пакета. Более старые пакеты ядра удаляются.

При использовании параметра color=never команда yum не использует в терминале никаких цветов. Если вы хотите работать с цветом, задавайте здесь параметр color=always.

Создание репозиториев. Каждый репозиторий определяется в собственном REPO-файле в каталоге /etc/yum.repos.d. В следующем фрагменте показаны репозитории для получения основных пакетов Fedora:

```
# Файл /etc/yum.repos.d/fedora.repo
[fedora]
name=Fedora $releasever - $basearch
failovermethod=priority
mirrorlist=https://mirrors.fedoraproject.org/metalink?repo=fedora-
$releasever&arch=$basearch
enabled=1
metadata_expire=7d
gpgcheck=1
gpgkey=file:///etc/pki/rpm-gpg/RPM-GPG-KEY-fedora-$basearch
```

Адрес репозитория можно по выбору указать в абсолютной форме с помощью `baseurl=...` или с использованием `mirrorlist=...` в виде файла-зеркала. В этом файле расположен список зеркальных серверов. Yum самостоятельно выбирает одно из зеркал. В конфигурационном файле Yum заменяет переменные `$releasever`, `$arch` и `$basearch` номерами версии дистрибутива Linux с указанием ее архитектуры. Кратко рассмотрим происхождение этих трех переменных.

- `$arch` определяется с помощью команды `uname` (если быть точным — с использованием одноименной функции Python, основанной на `uname`) и возвращает, к примеру, `x86_64`, если мы имеем дело с процессором 64-Bit-Intel/AMD.
- `$basearch` — это базовая архитектура, лежащая в основе `$arch` (например, i386).
- `$releasever` получается из номера версии пакета `edhat-release` или `fedora-release` (можно также задать в `yum.conf` название другого пакета с помощью ключевого слова `distroverpkg`; номер версии этого пакета считается номером версии дистрибутива).

Переменная `metadata_expires` указывает, в течение какого срока остаются действительны скачанные метаданные. Yum сохраняет метаданные в кэше и не скачивает их повторно, если прежние метаданные еще не устарели. Так экономится время и объем закачек (некоторые файлы с метаданными просто огромны), но в этом случае Yum может проигнорировать изменения, недавно внесенные в репозиторий (при необходимости можно удалить локальные метаданные командой `yum clean metadata`, тогда при следующем запуске Yum придется заново считать метаданные всех пакетов).

Оптимальная настройка для `metadata_expires` отличается в зависимости от репозитория: если его содержимое обновляется редко или вообще не обновляется, лучше задать длительный промежуток. Напротив, при работе с пакетами обновлений желательно задать короткий период или вообще отказаться от этой настройки.

Запрет пакетов. Если вы хотите, чтобы Yum не контактировала с определенными пакетами и при появлении новой версии они также не обновлялись, вставьте в `yum.conf` или в файл `*.repo` репозитория строку вида `exclude` *`имя1 имя2 имя3`*. В названиях пакетов можно использовать джокерные символы, то есть допускается и такая запись: `exclude xemacs*`.

Плагины. Yum можно расширять с помощью плагинов — тогда функционал программы становится еще богаче. Плагины можно настроить в файлах каталога `/etc/yum/pluginconf.d`.

Presto. Один из интереснейших плагинов Yum называется Presto: он позволяет использовать для обновлений пакеты Delta-RPM. В результате объем загрузок радикально снижается (на 60–80 %!), но нагрузка на процессор при проведении обновлений выше.

Дистрибутивные обновления. В принципе с помощью Yum можно проводить и обновление дистрибутивов. Для этого сначала требуется вручную установить пакеты `fedora-release-n.noarch.rpm` и `fedora-release-n.noarch.rpm`, а затем выполнить команду `yum upgrade`. Надо отметить, что я не сторонник дистрибутивных обновлений. При этом обязательно что-то идет не так, независимо от того, насколько интеллектуальна система управления пакетами. Когда это только возможно, старайтесь проводить новую установку.

Примеры

Управление пакетами целиком осуществляется с помощью команды `yum`. Синтаксис этой команды понятен из следующего примера. Приведенные далее команды показывают, как работает программа (вывод сокращен ради экономии места).

```
root# yum check-update
...
acl.x86_64          2.2.49-6.fc16      updates
cifs-utils.x86_64   4.6-1.fc16         updates
gdb.x86_64          7.1-32.fc16        updates
glx-utils.x86_64    7.8.1-8.fc16       updates

...
root# yum update
...
Vorgangsübersicht
   Installieren    0 Paket(e)
   Upgrade         35 Paket(e)
   Gesamte Downloadgrösse: 26 M
Ist dies in Ordnung? [j/N] : j
...
root# yum install mysql-server
...
Installieren:
 mysql-server               x86_645.1.48-2.fc16   updates   8.1 M
Installiert für Abhängigkeiten:
 mysql                   x86_645.1.48-2.fc16      updates   889 k
 perl-DBD-MySQL          x86_644.016-1.fc16       updates   135 k
 perl-DBI                x86_641.609-4.fc16       fedora    707 k

Vorgangsübersicht
 Installieren   4 Paket(e)
 Upgrade        0 Paket(e)
 Gesamte Downloadgrösse: 9.8 M
 Installed size: 28 M
Ist dies in Ordnung? [j/N] : j
```

```
root# yum history
ID     | Login user       | Date and time    | Action(s)  | Altered
-------------------------------------------------------------------------------
    12 | kofler <kofler>  | 2011-08-09 10:38 | Install    | 3
    11 | kofler <kofler>  | 2011-08-09 10:35 | Install    | 1
    10 | System <unset>   | 2011-08-06 12:47 | Install    | 2
...

root# yum history info 10
Transaction ID  : 10
Begin time      : Fri Aug 6 12:47:08 2011
Begin rpmdb     : 1239:cd1de7f23aa89953e4eeb9fa5e561f7fe36e079a
End time        :          12:47:22 2011 (14 seconds)
End rpmdb       : 1241:bad14ec1137d77b9d69ffa13ddf55138972644aa
User            : System <unset>
Return-Code     : Success
Transaction performed with:
    Installiert     rpm-4.8.1-2.fc16.x86_64
    Installiert     yum-3.2.27-4.fc16.noarch
    Installiert     yum-plugin-fs-snapshot-1.1.27-2.fc16.noarch
    Installiert     yum-presto-0.6.2-1.fc16.noarch
Packages Altered:
    Dep-Install     freeglut-2.6.0-5.fc16.x86_64
    Install         mesa-demos-7.8.1-6.fc16.x86_64
history info
```

Если вы впервые выполняете команду `yum`, система скачивает метаинформацию по всем созданным репозиториям. В дальнейшем эти файлы регулярно обновляются.

Группы пакетов

Программа Yum может распределять пакеты по группам, чтобы проще было установить все пакеты, необходимые для решения определенной задачи. Список доступных групп пакетов выводится командой `yum grouplist`, а `yum groupinfo name` показывает, какие пакеты относятся к одной и той же группе. Команда `yum groupinfo` подразделяет все пакеты на три категории: `mandatory` (обязательные), `default` (по умолчанию) и `optional` (необязательные). Выполнив `yum groupinstall` *имя*, вы установите все пакеты групп mandatory и default. Команда `yum` не содержит параметров, которые позволили бы автоматически установить и необязательные пакеты. Если вам нужна такая возможность, внесите в файл `yum.conf` следующие изменения:

```
# Дополнение в /etc/yum.conf
group_package_types = mandatory default optional
```

Если хотите обновить или удалить группу пакетов, используйте `yum groupupdate` или `yum groupremove` соответственно.

Пакеты с исходным кодом

Сама по себе команда `yum` не может устанавливать пакеты с исходным кодом. Эту задачу выполняет команда `yumdownloader` из пакета `yum-utils`. Следующая команда загружает пакет с исходным кодом редактора gedit в локальный каталог. При этом

активируются источники source, находящиеся в файлах *.repo и по умолчанию неактивные.

```
user$ yumdownloader --source gedit
```

Автоматические загрузки и обновления

Если установлен пакет yum-updatesd, то при запуске компьютера процесс Init-V активирует одноименную программу yum-updatesd.

Команда yum-updatesd позволяет регулярно проверять, есть ли доступные обновления. В зависимости от того, какие настройки сделаны в /etc/yum/yum-updatesd.conf, пакеты обновлений также загружаются и даже устанавливаются автоматически. В следующем листинге показана стандартная конфигурация Fedora 13, при которой автоматические обновления *не* производятся. Если вы хотите производить автоматические обновления, в трех случаях измените no на yes:

```
# /etc/yum/yum-updatesd.conf
[main]
# Проверять раз в час, доступны ли новые обновления
run_interval = 3600

# Связываться с сервером обновлений не реже, чем каждые 10 минут
updaterefresh = 600

# Осуществлять локальное оповещение об уведомлениях через dbus
emit_via = dbus
dbus_listener = yes

# Автоматически загружать обновления
do_download = no

# Загружать для обновления зависимые пакеты
do_download_deps = no

# Автоматически установить обновления
do_update = no
```

К сожалению, если программа обновлений Yum будет работать постоянно, это иногда может приводить к проблемам, связанным с блокировкой. Временно остановите yum-updatesd, прежде чем приступить к управлению пакетами. Не забудьте потом снова запустить yum-updatesd!

```
root# /etc/init.d/yum-updatesd stop
root# ... вручную установить или удалить пакеты ...
root# /etc/init.d/yum-updatesd start
```

Yum Extender (Yumex)

Yumex — это простой и удобный графический пользовательский интерфейс для Yum. При запуске Yum обновляет локальные метаданные по всем репозиториям.

Кроме того, вы можете найти пакеты, которые необходимо установить, отметить их для установки и, наконец, установить (кнопка Обработать очередь) (рис. 12.1). Теперь Yumex интерпретирует все взаимозависимости пакетов и выведет обобщающее диалоговое окно. Только после вашего подтверждения пакеты действительно будут скачаны и установлены.

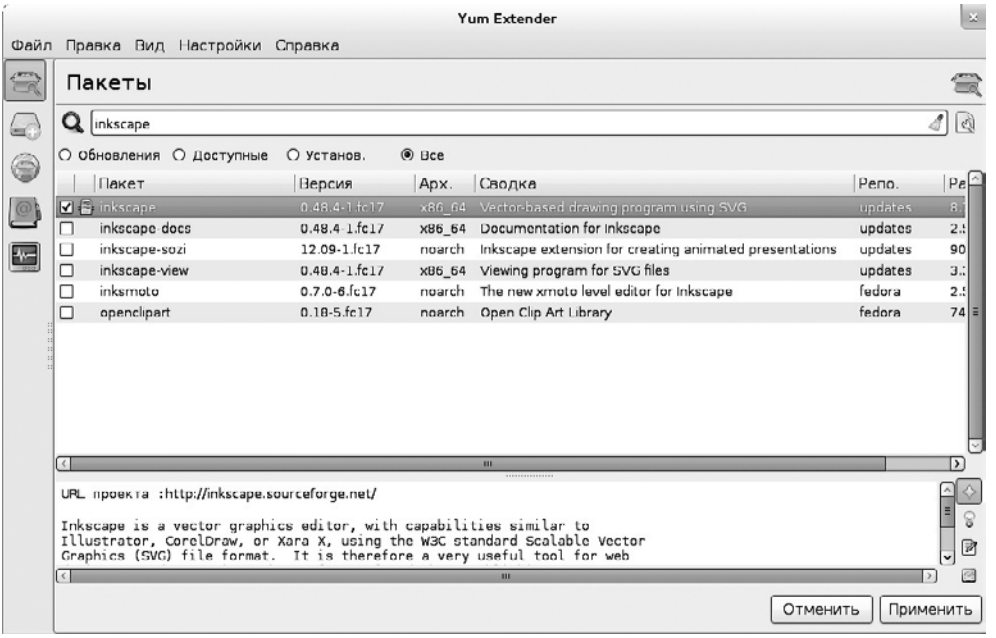


Рис. 12.1. Управление пакетами с помощью Yum Extender

12.3. ZYpp

В SUSE, как и в Fedora и Red Hat, используются RPM-пакеты. ZYpp, система управления пакетами, основанная на Yum, была разработана именно для Novell/SUSE. ZYpp означает «ZENworks, YaST, пакеты и патчи», причем использование Zenworks не является обязательным и предусмотрено только в корпоративных дистрибутивах Novell/SUSE. Подробная информация по ZYpp имеется на сайте http://en.opensuse.org/ZYpp.

Библиотека libzypp

За кулисами программы работает библиотека `libzypp`, предоставляющая основные функции ZYpp. Данная библиотека разбирается с репозиториями как YaST, так и YUM. Все файлы конфигурации, базы данных и кэша находятся в каталоге `/var/lib/zypp`. YaST и PackageKit при работе в SUSE обращаются к библиотеке `libzypp`.

ПРИМЕЧАНИЕ

Zypper различает обновления и патчи. Обновления — это пакеты, для которых доступна более новая версия, чем та, что установлена у вас. Патчи, в свою очередь, — это дополнительные пакеты (Delta-RPM).

SUSE использует для обновления пакетов патчи в форме Delta-RPM. Внешние репозитории, например Packman, предлагают новые версии пакетов в форме обновлений.

Репозитории

Репозитории сохраняются в текстовых файлах каталога `/etc/zypp/repos.d`. Если вы изменяете эти файлы в текстовом редакторе, не забудьте дополнительно удалить и все резервные копии! В противном случае в вашем списке репозиториев появятся двойники. В следующих строках дается определение свободно распространяемого репозитория для openSUSE 12.1:

```
# Файл /etc/zypp/repos.d/repo.oss.repo
[repo-oss]
name=openSUSE-12.1-Oss
enabled=1
autorefresh=1
baseurl=http://download.opensuse.org/distribution/12.1/repo/oss/
type=yast2
keeppackages=1
```

Интерфейс zypper

Это командный интерфейс для `libzypp`. Таким образом, `zypper` — аналог SUSE для `aptget` или `yum`. С его помощью можно искать, устанавливать, обновлять и удалять пакеты, а также управлять репозиториями. Команда `zypper` должна выполняться администратором. Более подробная информация приводится на сайте http://de.opensuse.org/Zypper/Anleitung.

В следующих примерах демонстрируется применение `zypper`. Первая команда перечисляет репозитории, вторая обновляет источники, третья устанавливает редактор nano, а четвертая определяет, какие есть обновления.

```
root# zypper repos
# | Псевдоним       | Имя                   | Включено   | Обновить
--+-----------------+-----------------------+------------+--------
1 | repo-non-oss    | openSUSE-12.1-Non-Oss | Yes        | Yes
2 | repo-oss        | openSUSE-12.1-Oss     | Yes        | Yes
...

root# zypper refresh
Все репозитории были обновлены.

root# zypper install nano
Будет установлен следующий НОВЫЙ пакет:
   ctags emacs emacs-info emacs-x11 fribidi libotf0 m17n-db m17n-lib xaw3d
Необходимо установить 9 пакетов.
```

```
Общий объем загрузок: 21,0 Мбайт. После операции будет использовано еще 85,5 Мбайт.
Продолжить? [y/n/?]: y

root# zypper list-updates
...
S | Репозиторий | Имя        | Версия     | Категория     | Статус
--+-------------+------------+------------+---------------+-------
v | Обновления  | bind-libs  | 9.7.1-1.5  | 9.7.1-2.1.1   | x86_64
v | Обновления  | bind-utils | 9.7.1-1.5  | 9.7.1-2.1.1   | x86_64
v | Обновления  | ethtool    | 6-84.1     | 6-85.1.1      | x86_64
...
```

Группы пакетов. Чтобы установить пакеты, необходимые для решения определенной задачи, например для применения компьютера в качестве файлового сервера, ZYpp использует так называемый *шаблон* (pattern). Команда `zypper search -t` *шаблон* выдает список всех таких групп пакетов. Команда `zypper info -t` *шаблон имя* показывает, какие пакеты относятся к группе *имя*. С помощью `zypper install -t` *шаблон имя* устанавливаются все пакеты группы *имя*.

History. В файле `/var/log/zypp/history` содержится исключительно полезная информация о том, когда какой пакет был установлен или удален из какого репозитория и какие конфигурационные работы были при этом проведены.

Обновления дистрибутивов. С помощью `zypper dup` вы можете, как в Debian или Ubuntu, обновлять дистрибутив, не останавливая его работу:

```
root# zypper updates     (Обновление имеющейся версии)
root# ...                (Замена версии репозиториев)
root# zypper dup         (Замена старых пакетов новыми)
root# reboot             (Новый запуск)
```

12.4. Управление пакетами Debian (команда dpkg)

Управление пакетами Debian осуществляется на двух уровнях: в этом разделе описана команда `dpkg`, которая отвечает за установку пакетов и управление ими на нижнем уровне. Данную команду можно сравнить с `rpm`. Команда может устанавливать отдельные пакеты, обновлять их, удалять и при этом тестировать, выполняются ли все взаимозависимости пакетов. Но эта команда не может самостоятельно отменять невыполненные взаимозависимости пакетов либо загружать информацию из репозиториев.

Выполнением этих задач занимается APT (Advanced Package Tool, усовершенствованное средство управления программными пакетами). Оно основано на `dpkg` и функционально напоминает описанные выше системы Yum и ZYpp. Собственно для управления пакетами предлагаются две команды: `apt-get` обычно применяется в Ubuntu, а `aptitude` считается основным инструментом для установки пакетов в Debian 5.0 и выше.

Если рассмотреть этот и следующий разделы с точки зрения установки пакетов в Debian, то будут выполняться все правила, действующие для других дистрибу-

тивов Linux, использующих этот формат пакетов. Кроме Debian это, например, дистрибутивы семейства Ubuntu — Mepis, Mint и Knoppix. Если вы переходите с дистрибутива на основе RPM на дистрибутив с пакетами Debian, рекомендую вам посмотреть следующий сайт — там дан хороший обзор команд `rpm`, а также эквивалентных им команд `dpkg` и `apt`: https://help.ubuntu.com/community/SwitchingToUbuntu/FromLinux/RedHatEnterpriseLinuxAndFedora.

В следующих примерах разъясняется использование команды в стандартных ситуациях. Кроме этого случая `dpkg` вряд ли пригодится вам на практике. Администрирование пакетов в большинстве дистрибутивов, использующих вариант пакетов для Debian, осуществляется непосредственно с помощью APT или основанного на нем инструмента.

```
root# dpkg --install test.deb
...
Распаковка теста ...
Создание теста ...

root# dpkg --search /etc/sensors3.conf
libsensors4: /etc/sensors3.conf

root# dpkg --listfiles libsensors4
...
/usr/lib
/usr/lib/libsensors.so.4.2.1
/etc
/etc/sensors3.conf
/etc/sensors.d
...
```

Статус пакета. Команда `dpkg --list` возвращает список всех установленных пакетов. При этом отображается код состояния, состоящий из одной-трех букв. Первая буква указывает желаемое состояние (`u` = `unknown`, `i` = `install`, `r` = `remove`, `p` = `purge`, `h` = `hold`), вторая буква — фактическое состояние (`n` = `not`, `i` = `installed`, `c` = `config files`, `u` = `unpacked`, `f` = `failed config`, `h` = `half installed`), третья — код ошибки (`h` = `hold`, `r` = `reinstall required`, `x` = `hold + reinstall required`).

```
root# dpkg --list | grep hplip
ii hplip        3.10.6-0ubuntu1 HP Linux Printing and Imaging System (HPLIP)
ii hplip-cups   3.10.6-0ubuntu1 HP Linux Printing and Imaging - CUPS Raster driver
ii hplip-data   3.10.6-0ubuntu1 HP Linux Printing and Imaging - data files
```

Чаще всего встречаются статусные коды `ii` (установленный пакет) и `rc` (пакет удален, но данные о его конфигурации еще доступны). Чтобы полностью удалить пакет rc, выполните команду `dpkg--purge` *`имя`*.

Статус `hold` означает, что пакет не должен изменяться, когда скачиваются обновления к нему. Две следующие команды показывают, как перевести пакет в состояние `hold` или снова отменить этот статус:

```
root# echo "имя_пакета hold"   | dpkg --set-selections
root# echo "имя_пакета install | dpkg --set-selections
```

Более подробные сведения по статусам пакетов и устранению связанных с этим проблем дает справка `man dpkg`.

Get-selections и set-selections. Чтобы быстро получить отсортированный список всех установленных пакетов, выполните `dpkg --get-selections`. Полученный таким образом список содержит меньше подробной информации, чем `dpkg --list`, и поэтому он гораздо нагляднее. Если вы переадресуете список и сохраняете его в текстовом файле, то можно позже установить пакеты на другом компьютере с помощью `dpkg --set-selections`.

```
root# dpkg --get-selections
accountsservice             install
acl                         install
acpi-support                install
acpid                       install
...
```

Новая конфигурация. В некоторых пакетах кроме автоматических конфигурационных сценариев имеются установочные инструменты, позволяющие сконфигурировать каждую отдельную программу. Если позже вы захотите изменить конфигурацию, выполните команду `dpkg-reconfigure` *`имя_пакета`*.

Метаданные. Команда `dpkg` располагает подробной метаинформацией по всем пакетам (описание пакета, список всех файлов пакета, данные о взаимозависимости и т. д.). Эти файлы предлагаются в формате DCTRL (Debian Control). В пакете `dctrl-tools` содержатся различные команды, позволяющие производить запросы данных DCTRL. В справке `man grep-dctrl` приводится подробное описание этой команды и множество конкретных примеров применения.

12.5. APT

APT для пакетов Debian так же важен, как и Yum для пакетов RPM: это высокоуровневая система управления пакетами, самостоятельно скачивающая пакеты из репозиториев и автоматически отменяющая межпакетные взаимозависимости. Комбинация из пакетов Debian и инструмента APT на настоящее время представляется наиболее проработанной системой управления пакетами в Linux. Среди прочего, она применяется в Ubuntu и Debian в качестве стандартной системы управления пакетами.

Для управления пакетами предлагаются две различные команды: `apt-get` и `aptitude`. Обе команды очень похожи друг на друга и при всей схожести обслуживания даже имеют практически одинаковый синтаксис. Команды `apt-get install` *`имя_пакета`* или `aptitude install` *`имя_пакета`* загружают указанный пакет и все пакеты, зависящие от него, и устанавливают их.

В настоящее время `apt-get` по умолчанию применяется в Ubuntu, а в Debian версии 5 и выше рекомендуется использовать `aptitude`. Неважно, работаете вы с Debian или с Ubuntu, — *обе* команды устанавливаются по умолчанию.

Кроме того, существуют версии APT для пакетов RPM, которые, однако, распространены не так широко, как Yum. В этом разделе будет рассмотрен только APT

для пакетов Debian. Информация по APT находится на следующем сайте: http://apt-rpm.org/.

Конфигурация

Конфигурация APT производится в двух файлах — `apt.conf.d/*` и `sources.list`, находящихся в каталоге `/etc/apt`. Более подробная информация по репозиториям может храниться в каталоге `sources.list.d`.

`apt.conf.d/*` как правило, содержит лишь немногие базовые настройки, которые обычно нужно оставить в том виде, в котором они заданы по умолчанию (также см. `man apt.conf`). Файл `sources.list` (справка `man sources.list`) уже интереснее. Этот файл включает в себя построчное описание репозиториев APT. Синтаксис каждой строки таков:

```
pakettyp uri distribution [компонент1] [компонент2] [компонент3] ...
```

Тип обычных пакетов Debian называется `deb`, а для пакетов с исходным кодом — `deb-src`. Наряду с каталогами HTTP и FTP APT поддерживает обычные каталоги, RSH и SSH-серверы, а также CD и DVD.

Правда, случай с репозиториями CD и DVD особый: такие пакеты создаются командой `apt-cdrom`, описанной чуть ниже. Просто добавить строку `deb cdrom` недостаточно.

В третьем столбце стоит название дистрибутива (ведь на сервере могут находиться пакеты для нескольких дистрибутивов или их версий). Во всех остальных столбцах указываются компоненты дистрибутива, которые можно учитывать. Названия компонентов зависят от конкретного дистрибутива и пакета! Например, в Ubuntu различаются пакеты `main`, `restricted`, `universe` и `multiverse`, а в Debian — `main`, `contrib`, `non-free` и т. д.

Версии пакетов, названные раньше, считаются предпочтительными: то есть если пакет доступен для скачивания в нескольких источниках, APT скачивает его по первой ссылке. в следующих строках показан синтаксис. Это строки из немецкой версии Ubuntu 12.10.

```
# Файл /etc/apt/sources.list
deb http://de.archive.ubuntu.com/ubuntu/ quantal          main restricted universe
multiverse
deb http://de.archive.ubuntu.com/ubuntu/ quantal-updates  main restricted universe
multiverse
deb http://security.ubuntu.com/ubuntu    quantal-security main restricted universe
multiverse
```

Изменения в `sources.list` лучше всего вносить в текстовом редакторе. Если не хотите пользоваться редактором, попробуйте графический пользовательский интерфейс, например Synaptic.

Декларирование CD в качестве репозитория. В качестве источников данных для APT также можно использовать CD. В таком случае для каждого диска необходимо выполнить команду `apt-cdrom add`. Она считывает метаданные APT компакт-диска и заносит доступные пакеты в файл кэша. Кроме того, обновляется `sources.lst`.

Если apt-cdrom жалуется на то, что не может найти CD/DVD, укажите каталог с параметром -d.

```
root# apt-cdrom -d /media/dvd add
```

Установка APT-ключа. В большинстве APT-источников в Интернете метафайлы для описания репозиториев зашифрованы криптографическими ключами. Кроме того, файлы-оглавления APT содержат контрольные суммы для всех пакетов. Такие механизмы контроля позволяют обеспечить постепенное изменение пакета. Однако этот механизм контроля действует лишь в том случае, если APT известна общедоступная часть ключа, позволяющая определить происхождение пакета. Чтобы настроить ключ для APT, пользуйтесь командой apt-key:

```
root# apt-key add кодовый_файл.gpg
```

Команда apt-get

Само управление пакетами производится командой apt-get или aptitude. Синтаксис apt-get понятен на следующих примерах. Показанный в них вывод сокращен ради экономии места.

Обновление информации по APT. Перед установкой пакетов выполните apt-get update, загрузив таким образом новейшую информацию из репозиториев. Пакеты при этом ни устанавливаются, ни обновляются. В данном случае речь идет только об описаниях пакетов!

Установка пакетов. Дополнительно выполните apt-get install — при этом необходимо указать правильное название пакета. Если команда обнаружит, что определенные взаимозависимости пакетов не соблюдаются, она также предложит установить недостающие пакеты. Когда вы примете это предложение, apt-get скачает файлы пакетов и установит их. В следующем примере apt-get рекомендует установить несколько дополнительных пакетов.

```
root# apt-get update
...
root# apt-get install apache2
...
Устанавливаются следующие дополнительные пакеты:
apache2-mpm-worker apache2-utils apache2.2-common libapr1 libaprutil1
Предлагаемые пакеты:
apache2-doc apache2-suexec apache2-suexec-custom
Устанавливаются следующие НОВЫЕ пакеты:
apache2 apache2-mpm-worker apache2-utils apache2.2-common libapr1 libaprutil1
0 обновлено, 6 новых установлено, 0 к удалению и 11 не обновлено.
Необходимо скачать архивы общим размером 1472 Кбайт.
После этой операции на диске будет занято еще 5452 Кбайт места.
Хотите продолжить? [Y/n]?
...
```

Удаление пакетов. Команда apt-get remove *имя_пакета* удаляет указанный пакет. Пакеты, которые первоначально были загружены с удаляемым пакетом и связаны с ним зависимостями, остаются при этом нетронутыми. В данном случае используйте apt-get autoremove. Эта команда удаляет все пакеты, которые больше не нужны.

Обновление пакетов. Команда `apt-get dist-upgrade` обновляет все установленные пакеты. Если изменились межпакетные взаимосвязи и требуется установить новые пакеты или удалить старые, эти операции также выполняются.

```
qqroot# apt-get upgrade
...
Следующие пакеты будут обновлены:
   libmozjs1d xulrunner-1.9 xulrunner-1.9-gnome-support
3 обновлено, 0 новых установлено, 0 к удалению и 0 не обновлено.
Необходимо скачать архивы общим размером 8208 Кбайт.
После этой операции на диске будет занято еще 8192 Кбайт места.
Хотите продолжить? [Y/n]? Y
...
```

У команды `apt-get dist-upgrade` есть еще вариант `upgrade`: он также обновляет все пакеты. Разница заключается в том, что при этом не устанавливаются никакие новые или имеющиеся пакеты.

Инсталляция исходного кода. Команда `apt-get source` *имя_пакета* устанавливает исходный код нужного пакета в текущий каталог.

Программа aptitude

Текстовая программа `aptitude` также построена на основе APT. Если вы используете эту программу в командном режиме (`aptitude install` *имя_пакета*), ее синтаксис во многом совместим с `apt-get`. Кроме того, эту программу можно использовать в консоли — с текстовым пользовательским интерфейсом (рис. 12.2). Для этого просто запустите программу без дополнительных параметров. Для перехода в меню нажмите клавишу F10.

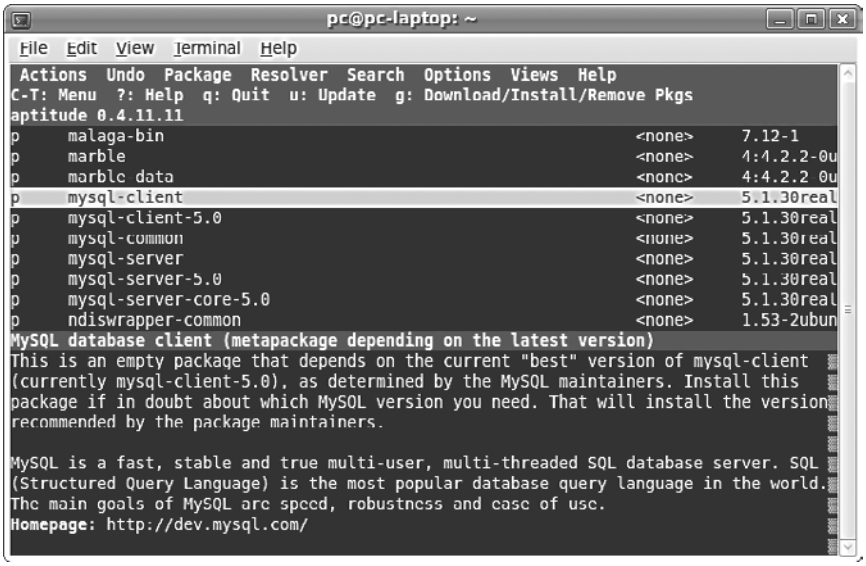


Рис. 12.2. Управление пакетами с помощью программы aptitude

По сравнению с apt-get программа aptitude имеет принципиальное преимущество: она отмечает, какие зависимые пакеты были установлены, и автоматически удаляет их при деинсталляции. Если, например, вы устанавливаете программу *xyz*, для работы которой требуется пять дополнительных пакетов (lib-abc, lib-efg и т. д.), эти пакеты удаляются (если от них не зависит какой-нибудь другой пакет).

Если же вы удаляете *xyz* с помощью aptget или Synaptic, зависимые пакеты lib-abc, lib-efg и т. д. остаются в системе. Пройдет немного времени, и никто уже не вспомнит, зачем были установлены эти пакеты.

В Debian для управления пакетами специально рекомендуется использовать aptitude вместо apt-get.

Пример. Две следующие команды сначала устанавливают пакет html2ps, а потом снова его удаляют. Отмечу, что большинство дополнительных пакетов, установленных вместе с html2ps, снова удаляются. Большинство, но не все — команда aptitude несовершенна.

```
root# aptitude install html2ps
Считываются списки пакетов... Готово
...
Дополнительно устанавливаются следующие пакеты:
   gs-gpl html2ps libcompress-zlib-perl libfont-afm-perl libhtml-format-perl
   libhtml-parser-perl libhtml-tagset-perl libhtml-tree-perl libmailtools-perl
...
Хотите продолжить? [Y/n/?] Y
...
root# aptitude remove html2ps
...
Следующие пакеты не используются и будут УДАЛЕНЫ:
   libfont-afm-perl libhtml-format-perl libhtml-parser-perl libhtml-tagset-perl
   libhtml-tree-perl libmailtools-perl libtimedate-perl liburi-perl libwww-perl
   perlmagick
...
```

Команда tasksel

Как правило, для установки групп пакетов в Debian и Ubuntu используются метапакеты: это пустые пакеты, определяющие только отдельные межпакетные взаимосвязи. Например, вместе с метапакетом build-essential инсталлируется несколько пакетов с важными инструментами разработки (Compiler, make и т. д.).

Есть еще один способ определения групп пакетов — на основе команды tasksel. Этот механизм предназначен в первую очередь для того, чтобы можно было с легкостью выбирать пакеты при установке дистрибутива. Разумеется, tasksel можно использовать и позже, при эксплуатации компьютера. Список всех имеющихся групп пакетов выводится командой tasksel --list-task. Для установки групп пакетов применяется команда tasksel install groupname. Если tasksel выполняется без дополнительных параметров, то система выводит диалоговое окно, в котором можно выбрать требуемые группы пакетов.

Команда apt-cache

Команда apt-cache узнает различные данные относительно доступных или уже установленных пакетов:

```
root# apt-cache show apache2
Package: apache2
Priority: optional
Section: web
...
Description: передовой, масштабируемый, расширяемый веб-сервер
  Apache v2 — представитель следующего поколения вездесущего веб-сервера
  Apache. Данная версия — полностью переработанная — содержит множество
  нововведений, в частности функцию обработки потоков, новый интерфейс API,
  поддержку IPv6, фильтрацию запросов и откликов и многое другое.
...

root# apt-cache search scribus | sort
lprof - Hardware Color Profiler
scribus - Open Source Desktop Publishing
scribus-template - additional scribus templates
```

Автоматизация обновлений

Как правило, после установки APT требуется откорректировать некоторые конфигурационные файлы, чтобы система могла автоматически скачивать обновления, а при необходимости и устанавливать их. Нужные конфигурационные файлы входят в состав пакета unattended-upgrades, который по умолчанию устанавливается в Debian и Ubuntu.

Для автоматизации скачивания и установки обновлений была разработана программа Cron, которая ежедневно выполняет сценарий /etc/cron.daily/apt. Этот сценарий интерпретирует конфигурационный файл /etc/apt/apt.conf.d/* и при необходимости выполняет команду обновления unattended-upgrade. Чтобы активизировать автоматические обновления, нужно вручную внести в конфигурационные файлы следующие изменения:

```
// Файл /etc/apt/apt.conf.d/10periodic
// Активизировать ежедневные обновления
APT::Periodic::Unattended-Upgrade «1»;

// Файл /etc/apt/apt.conf.d/50unattended-upgrades
// Проводить стандартные обновления и обновления системы безопасности
Unattended-Upgrade::Allowed-Origins {
      "${distro_id} stable";
      "${distro_id} ${distro_codename}-security";
      "${distro_id} ${distro_codename}-updates";
};

// Не обновлять следующие пакеты
Unattended-Upgrade::Package-Blacklist {
```

```
        // "vim";
        // ...
};

// Прислать электронное сообщение о статусе обновления
// Unattended-Upgrade::Mail "root@localhost";
```

`APT::Periodic::Unattended-Upgrade` указывает, с каким интервалом (в днях) система должна пытаться скачивать обновления.

С помощью `Allowed-Origins` укажите, какие репозитории должны учитываться при проведении обновлений. С использованием `Package-Blacklist` можно автоматически исключить определенные пакеты при проведении обновлений.

Наконец, команда `mail` позволяет указать адрес, на который команда `unattended-upgrade` посылает краткое статусное уведомление после того, как обновление будет успешно завершено. Для рассылки электронной почты необходимо настроить на компьютере почтовый сервер (MTA) и дополнительно установить пакет `mailx`.

В заключение скажу еще несколько слов о процессе обновления: в каталоге `/etc/cron.daily/apt` содержится команда `sleep`, задерживающая выполнение сценария на случайное количество секунд (до 1800). Такая принудительная пауза позволяет избежать ситуации, в которой тысячи компьютеров, использующих cron, одновременно обращались бы к одним и тем же репозиториям.

Сценарий `/usr/bin/unattended-upgrade` протоколирует все обновления или попытки обновлений в файлах регистрации в каталоге `/var/log/unattended-upgrade`. Кроме того, `/etc/cron.daily/apt` закладывает в каталоге `/var/lib/apt/periodic/` файлы с отметками времени, показывающими, когда именно в последний раз проводилась определенная операция.

Команда `unattended-upgrade` не обновляет те пакеты, в которых имеется так называемая инструкция conffile prompt, то есть конфигурационные файлы которых были обновлены вручную. К сожалению, эти данные можно узнать только из файлов регистрации, но не из статусного сообщения, пришедшего по электронной почте.

Поскольку на практике такие изменения проводятся довольно часто, несмотря на автоматические обновления, вам придется регулярно проверять, появились ли обновления, которые необходимо установить вручную.

Обновления ядра вступают в силу только после перезагрузки компьютера. Команда `unattended-upgrade` этим не занимается, то есть вы сами должны перезапустить компьютер командой `reboot`.

СОВЕТ

Автоматические обновления хороши тем, что снижают риск проникновения злоумышленника в случайную брешь в системе безопасности вашего компьютера. Если из-за такого неправильного обновления некоторые функции сервера перестанут работать, то последствия могут быть катастрофическими.

Альтернатива автоматических обновлений — сценарий Cron, ежедневно проверяющий, доступны ли свежие обновления (например, с помощью команды apt-get dist-upgrade –simulate), и пересылающий результат администратору по электронной почте.

Обновления версий или дистрибутивов

С помощью команды `apt-get dist-upgrade` можно обновить целый дистрибутив до новейшей версии. Для этого потребуется соответствующим образом заново настроить репозитории в `/etc/apt/sources.list`.

В Ubuntu обновление дистрибутива можно произвести с помощью команды `do-release-update -m desktop` (для ПК) или `-m server` (для сервера). В версиях Ubuntu LTS (рассчитанных на долговременную поддержку) новые релизы предусмотрены лишь для следующих версий, то есть, например, 8.04 или 12.04. Если вы хотите обновить версию, которая не рассчитана на долговременную поддержку, вам сначала нужно в каталоге `/etc/updatemanager/release-upgrades` установить для переменной `Prompt` значение `normal` вместо `lts`.

Несмотря на то что в Debian хорошо проработана система управления пакетами, обновление версий этой системы — дело очень деликатное. Если после обновления все программы и серверные службы будут работать как надо, то это счастливое исключение, а не правило.

Буфер обмена пакетов

Если в вашей сети работают десятки компьютеров с Debian и Ubuntu и каждый обращается за обновлениями к внешнему репозиторию, то общий ежемесячный трафик достигает нескольких гигабайт. Даже если тарифы на пользование Интернетом безлимитные, обновления засоряют сеть и чрезмерно замедляют передачу данных по LAN. В данном случае кажется очевидным, что не помешает организовать центральный буфер обмена (прокси), из которого все компьютеры будут брать пакеты для обновления. Для решения этой задачи существует большое количество программ, в частности `apt-cacher`, `apt-cacher-ng`, `squid-deb-proxy`, `apt-proxy` (уже не поддерживается) и `approx`. В качестве примера я расскажу о программе `apt-cacher`.

Серверная конфигурация

На кэш-сервере (то есть на компьютере, управляющем буфером обмена) необходимо установить пакет `apt-cacher`. (Из многочисленных руководств по `apt-cacher`, имеющихся в Интернете, можно сделать вывод, что вам также потребуется установить Apache. Это неверно — `apt-cacher` работает сам! Взаимодействие с Apache оправданно только в тех случаях, когда обмен информацией с `apt-cacher` должен производиться через HTTP-порт 80.)

```
root# apt-get install apt-cacher
```

Чтобы в дальнейшем пакет `apt-cacher` запускался автоматически как демон, внесите небольшие изменения в файл `/etc/default/apt-cacher`:

```
# Файл /etc/default/apt-cacher
AUTOSTART=1
...
```

Остальные параметры конфигурации содержатся в файле `/etc/apt-cacher/apt-cacher.conf`. Большинство установок можно оставить как есть, тогда APT-прокси

будет доступен всем компьютерам через порт 3142. Файлы пакетов будут сохраняться в каталоге /var/cache/apt-cacher. По соображениям безопасности рекомендуется внести в конфигурационный файл и следующие изменения:

```
# Файл /etc/apt-cacher/apt-cacher.conf
...
daemon_addr=192.168.0.1
allowed_hosts=192.168.0.0/24
...
```

Демон связывается с определенным адресом (в данном случае 192.168.0.1, этот показатель важен для компьютеров с несколькими интерфейсами) и позволяет пользоваться прокси только клиентам, относящимся к адресному пространству 192.168.0.*. После внесения в конфигурацию таких изменений мы в первый раз запускаем apt-cacher:

```
root# service apt-cacher start
```

Пакет apt-cacher заносит все случаи доступа, а также возможные ошибки в файлы регистрации, находящиеся в каталоге /var/log/apt-cacher.

Импорт имеющихся пакетов

Как правило, в каталоге /var/cache/apt/archive на сервере уже имеется множество пакетов, загружаемых локальной системой управления пакетами для установки либо обновления. Вы можете импортировать эти пакеты в кэш apt-cacher. Это целесообразно делать в тех случаях, когда ожидается, что данные пакеты потребуются другим компьютерам, работающим в локальной сети:

```
root# cd /usr/share/apt-cacher
root# ./apt-cacher-import.pl /var/cache/apt/archives
```

Клиентская конфигурация

На клиентской машине необходимо включить браузер и убедиться, что apt-cacher доступен через HTTP. Для этого укажите следующие адреса (конечно же, при этом необходимо изменить мой apt-cacher на хост-имя вашего компьютера или прокси-сервера):

- http://mein-apt-cacher:3142;
- http://mein-apt-cacher:3142/report.

На первой странице обобщается конфигурация, а на второй сообщается информация об эффективности прокси, которая тем выше, чем дольше работает прокси и чем больше клиентов им пользуется.

Когда apt-cacher уже будет работать, вам лишь останется внести еще одно маленькое изменение в конфигурацию APT, чтобы прокси-сервером могла пользоваться и команда apt-get. Для этого потребуется создать следующий новый файл и изменить mein-apt-cacher на хост-имя компьютера либо адрес прокси-сервера:

```
// Файл /apt/apt.conf.d/01proxy
Acquire::http::Proxy "http://mein-apt-cacher:3142/";
```

Теперь все команды APT будут пользоваться новым прокси. Файлы, которых на нем еще нет, разумеется, нужно, как и раньше, скачивать из Интернета. Но если другой компьютер в сети также решит обновиться или установить пакет, который был незадолго до этого инсталлирован на ином компьютере, необходимые пакеты уже будут в распоряжении (не забудьте закомментировать первую строку, поставив символы //, — возможно, вы захотите поработать с ноутбуком в дороге и вам при этом понадобится установить обновления и пакеты, не имея доступа к прокси-серверу)!

Synaptic

Для APT существует множество графических пользовательских интерфейсов, и лучшая из таких программ на настоящий момент — Synaptic (рис. 12.3). Эта программа рассчитана, скорее, на опытных пользователей. В любом случае по умолчанию Synaptic не устанавливается ни в Ubuntu, ни в Debian 7. Canonical постепенно продвигает Ubuntu Software Center, а Debian в версии 7 и выше переходит на использование PackageKit.

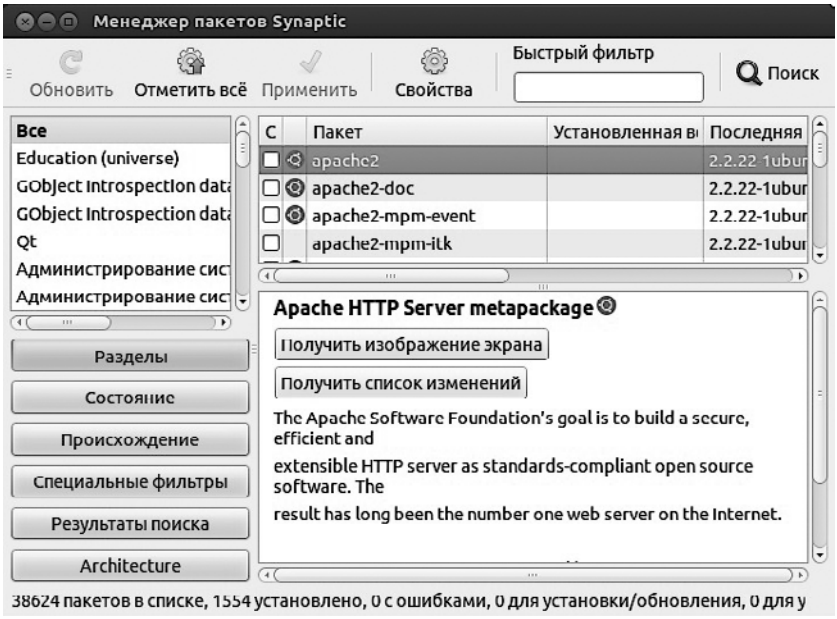


Рис. 12.3. Окно программы Synaptic

Поиск пакетов. Synaptic обладает сразу двумя поисковыми функциями: быстрым поиском по названиям и описаниям пакетов и обычным поиском, учитывающим другие критерии. Обе поисковые функции можно комбинировать друг с другом: тогда быстрый поиск будет производиться по результатам обычного поиска. Однако часто такая связь не нужна: удалите поисковый запрос в поле для быстрого поиска либо щелкните в столбце Поиск результатов на записи Все, чтобы вернуться к обычной функции поиска.

Установка. Если хотите установить определенный пакет, перейдите к инсталляции двойным щелчком кнопкой мыши. Если пакет зависит от другого пакета, то появляется диалоговое окно, где перечислены пакеты, которые также необходимо установить. Сама установка начинается после нажатия кнопки Принять, после чего вы должны одобрить все планируемые действия.

Если сначала нажать кнопку Настроить фильтр, а затем выбрать пункт Сохранить изменения в открывшемся меню, то программа выведет список всех пакетов, подготовленных для установки. Вы всегда можете отследить процесс, выполнив команду Файл ▸ История.

Управление репозиториями. Управлять репозиториями можно в диалоговом окне Настройки ▸ Репозитории. Здесь отображаются все известные репозитории. Установив флажки напротив определенных пакетов, вы можете быстро активизировать или деактивировать нужные вам пакеты. С помощью EDIT можно изменять свойства имеющихся репозиториев, а с использованием ADD — добавлять новые репозитории.

Блокировка. Одновременно может работать лишь одна программа управления пакетами. Если вы попытаетесь запустить сразу две такие программы, то система выдаст предупреждение: unable to get exclusive lock (невозможно обеспечить исключающую блокировку). Это означает, что программа не может обратиться к внутренним системным файлам, отвечающим за управление пакетами. Необходимо завершить обе конфликтующие программы.

Иногда предупреждение о блокировке выводится и тогда, когда на первый взгляд работает только одна программа управления пакетами. Причина обычно заключается в том, что при завершении предыдущей программы не был правильно удален lock-файл. При необходимости просто удалите его сами:

```
root# rm /var/lib/dpkg/lock
```

Дефектные пакеты. Иногда при инсталляции или деинсталляции пакета возникает проблема и процесс не удается правильно завершить. В результате пакет помечается как дефектный. Synaptic и другие инструменты для управления пакетами отказываются работать, пока эта проблема не будет решена.

Для устранения проблемы нажмите в Synaptic в списке страниц кнопку Специальные фильтры, а затем выберите пункт Дефектный. Тогда Synaptic отобразит список всех дефектных пакетов. Отметьте все пакеты, нажав сочетание Ctrl+A, щелкните в списке правой кнопкой мыши и выберите пункт контекстного меню Запретить для новой установки. Далее проведите установку заново, нажав кнопку Принять. Если опять возникнут проблемы, отметьте дефектные пакеты и удалите их.

12.6. PackageKit

PackageKit — это еще один пользовательский интерфейс для установки пакетов и управления ими. Важнейшая особенность этой программы заключается в том, что она совместима с многими системами управления пакетами, в том числе с APT, Yum и Zypper. PackageKit (либо его вариант для KDE — KPackageKit) применяется, среди прочего, в Fedora, Kubuntu и openSUSE, правда, в последней только для

выполнения обновлений в Gnome. Для получения прав администратора PackageKit обращается к PolicyKit (см. раздел 8.4).

Содержание и конфигурация системы. PackageKit состоит из нескольких частей, которые обычно содержатся в отдельных пакетах: к важнейшим компонентам относятся базовые функции или команды (пакет `packagekit`), интерфейс для работы с внутренней системой управления пакетами (например, `packagekitbackend-apt`) и графический пользовательский интерфейс (пакеты `gnome-packagekit` или `kpackagekit`). Названия пакетов отличаются в зависимости от дистрибутива.

PackageKit конфигурируется с помощью файлов из каталога `/etc/PackageKit/*`. Важнейшая настройка — это `DefaultBackend` в файле `PackageKit.conf`: она указывает, к какому *внутреннему интерфейсу* (backend) должна обращаться программа.

Для координирования операций PackageKit требуется демон `packagekitd`. При необходимости программа автоматически запускается командами PackageKit либо с соответствующих интерфейсов.

Используя команду `pkcon`, можно выполнять операции с пакетами с консоли либо автоматизировать их с помощью сценария. Обратите внимание, что `pkcon` нельзя выполнять от лица администратора! Ее нужно запускать, будучи обычным пользователем. Если потом вам нужно получить права администратора, можно запустить программу PolicyKit (это правило действует и для графических пользовательских интерфейсов). Если вы хотите отслеживать с консоли, чем занята PackageKit, выполните команду `pkmon`.

Установка пакетов. Далее будет описан графический интерфейс PackageKit для работы с Gnome — я пользовался версией для Fedora 11. Для установки дополнительных пакетов либо для удаления имеющихся запустите программу `gpk-application`. Теперь можно выполнить поиск по имени пакета (рис. 12.4) или пролистать разные группы взаимозависимых пакетов (Package collections).

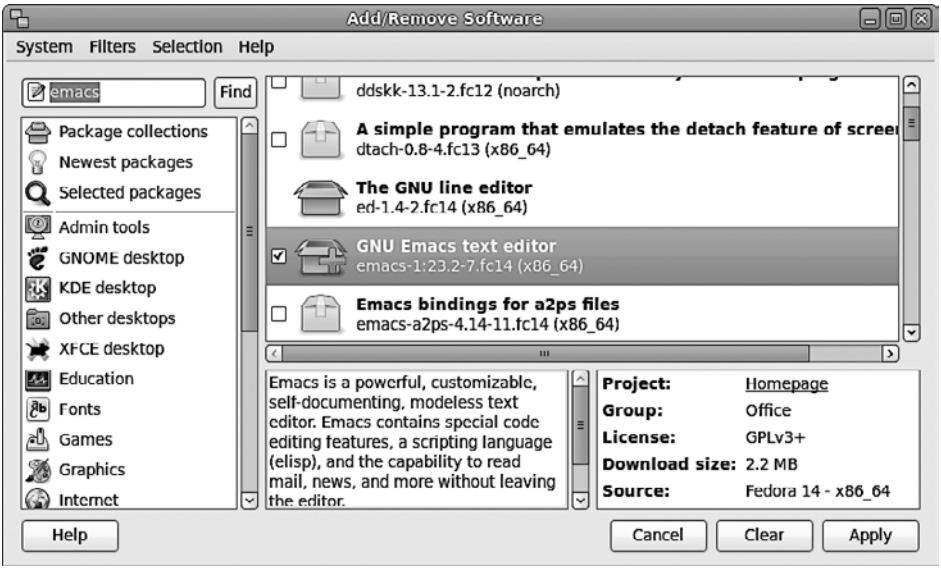


Рис. 12.4. Окно программы PackageKit, используемой для установки пакетов

В меню Filters (Фильтры) можно отфильтровать списки результатов, так как иногда они просто огромны. Например, можно отдельно отобразить уже установленные пакеты, пакеты для разработки и пакеты с графическим пользовательским интерфейсом. Нажатие кнопки Apply (Применить) запускает инсталляцию или деинсталляцию выбранного пакета.

Обновления. Программа `gpk-update-icon`, которая по умолчанию работает в Gnome, в тестовом режиме регулярно проверяет, доступны ли новые обновления. Если доступны, об этом сообщит значок на панели. Затем щелчком кнопкой мыши запускается программа `gpk-update-viewer`, выдающая подробную информацию о доступных обновлениях. После запуска процесса диалоговое окно состояния информирует вас о том, как протекает обновление.

12.7. TAR

Опытным пользователям Linux часто приходится устанавливать такое программное обеспечение, которое по форме не является пакетом для определенного дистрибутива. На различных серверах Linux в Интернете лежат гигабайты программ для Linux в архивах TAR.

Архивы, запакованные с помощью программы `gzip`, обычно имеют расширения `*.tgz` или `*.tar.gz`. Архив устанавливается на компьютере благодаря программе `tar`.

```
root# tar -tzf archiv.tar.gz          (Отображение содержимого архива)
root# tar -xzf archiv.tar.gz          (Распаковка файлов относительно текущего ка-
талога)
root# tar -xzf archiv.tar.gz "*.tex"  (Распаковка только файлов *.tex)
root# tar -xzf archiv.tar.gz -C dir   (Распаковка в один каталог)
```

Все чаще для создания архивов применяется новая, более мощная программа `bzip2`. Она распознает такие архивы по расширению `*.tar.bz2`. Чтобы отобразить или распаковать подобный архив, нужно использовать вместо `-z` параметр `tar -j`, то есть, например, `tar -tjf archiv.tar.bz2`.

Зачастую программы доступны только в виде исходного кода и перед применением их еще надо скомпилировать. Предполагается, что у вас должны быть инструменты разработки (`gcc`, `make` и т. д.), а также все необходимые библиотеки (devel-пакеты).

ВНИМАНИЕ

Если вы устанавливаете программные пакеты, распакованные из tar, они не учитываются ни одной из систем управления пакетами. Базы данных RPM не знают о программах, которые вы установили. По этой, а также по другим причинам всегда старайтесь устанавливать пакеты, созданные специально для вашего дистрибутива.

12.8. Преобразование одних форматов пакетов в другие

Что делать, если вам необходим пакет в формате RPM, но работать с ним предполагается в Debian или Ubuntu? Как поступить, если вы хотите просмотреть всего

один файл из пакета RPM? В таких случаях используйте команду alien. Она преобразует пакеты из одних форматов в другие (RPM, DEB, архив TAR и Stampede SLP).

К сожалению, alien безошибочно работает только с простыми пакетами. Если же в пакете используются установочные сценарии либо конкретный формат пакетов имеет другие специфические свойства, установка преобразованных пакетов часто завершается неудачно. В принципе, alien — это инструмент для профессионалов Linux.

Желаемый формат пакета задается с помощью параметров --to-deb (Debian), --to-rpm (RPM) или --to-tgz (архив TAR). Команда alien должна выполняться администратором, чтобы можно было правильно настроить пользователя и права доступа к новому пакету.

Следующая команда преобразует Debian в RPM:

```
root# alien --to-rpm paket.deb
```

Приведенные ниже команды показывают, как извлечь из RPM-пакета конкретный файл. Для этого пакет сначала преобразуется в архив TAR, затем с помощью команды tar из него извлекается файл и благодаря команде less этот файл отображается. Разумеется, вместо tar можно воспользоваться файловым менеджером Konqueror или архивными программами, например ark или file-roller. Они отображают содержимое архива в удобном виде.

```
root# alien --to-tgz paket.rpm
root# tar -xzf paket.tgz ./usr/share/doc/packages/paket/TODO
root# less ./usr/share/doc/packages/paket/TODO
```

12.9. Управление параллельными установками

В Linux часто можно выбирать из нескольких равнозначных программ, выполняющих одни и те же задачи, а иногда даже использующих одни и те же команды. Это печатные системы, редакторы, команды FTP, окружения Java и др. В некоторых ситуациях бывает целесообразно параллельно установить несколько вариантов или даже несколько версий одной и той же программы. Если при этом каждая программа будет находиться в собственном каталоге, то конфликты при установке не возникнут. Однако какая именно программа будет применяться для выполнения той или иной команды?

Для решения этого вопроса многие распространенные дистрибутивы используют концепцию, впервые появившуюся в Debian и основанную на символьных ссылках, находящихся в каталоге /etc/alternatives. В следующем списке указано, в каком пакете в разных дистрибутивах содержится каталог alternatives и соответствующая управляющая команда update-alternatives.

- Debian, Ubuntu — пакет dpkg;
- Red Hat, Fedora — пакет chkconfig;
- SUSE — пакет update-alternatives.

Эту концепцию лучше объяснить на примере. Предположим, что на одном компьютере установлены две версии Java. Программы Java выполняются с помощью класса `java`. В таком случае `/usr/bin/java` строится в виде ссылки на `/etc/alternatives/java`, а это, в свою очередь, еще одна ссылка, указывающая на нужную версию Java.

```
user$ ls -l /usr/bin/java
   /usr/bin/java -> /etc/alternatives/java
user$ ls -l /etc/alternatives/java
   /etc/alternatives/java -> /usr/lib/jvm/java-6-openjdk/jre/bin/java
```

Управление ссылками обычно происходит автоматически и осуществляется сценариями при установке пакета. При этом используется команда `update-alternatives`. В Red Hat и Fedora эта команда также доступна под названием `alternatives`.

Перечисление альтернатив

С помощью команды `update-alternatives --display` можно определить, какие версии определенной программы доступны и какая версия используется по умолчанию. В следующих строках показан результат для Java в Ubuntu 9.10 при установке двух версий Java. Строки `slave` (ведомые) относятся к командам, подчиненным данной программе, а также к страницам справки `man`. При изменении ссылки на команду программа `update-alternatives` автоматически обновляет все ведомые ссылки.

```
root# update-alternatives --display java
update-alternatives --display java
java - Auto-Modus
   Сейчас ссылка указывает на /usr/lib/jvm/java-6-openjdk/jre/bin/java
/usr/lib/jvm/java-6-openjdk/jre/bin/java — приоритет 1061
   Slave java.1.gz: /usr/lib/jvm/java-6-openjdk/jre/man/man1/java.1.gz
/usr/lib/jvm/java-6-sun/jre/bin/java — приоритет 63
   Slave java.1.gz: /usr/lib/jvm/java-6-sun/jre/man/man1/java.1.gz
В настоящее время "лучшая" версия /usr/lib/jvm/java-6-openjdk/jre/bin/java.
```

Обычно управление ссылками происходит автоматически: каждый установленный пакет получает указатель приоритета. Команда `update-alternative` при каждой (де)инсталляции активирует альтернативу с наивысшим приоритетом.

Альтернативы

Команда `update-alternatives -config` определяет вариант, который станет активным. Она выдает список альтернатив, предлагаемых на выбор, и из них нужно выделить один. Теперь `update-alternatives` обновит ссылки. При необходимости можно снова перейти в автоматический режим с помощью команды `update-alternatives -auto`.

```
root# update-alternatives --config java
Для альтернативы java существует два варианта на выбор
(они находятся в /usr/bin/java).
```

```
Вариант  Путь                                          Приоритет  Статус
-----------------------------------------------------------------
* 0      /usr/lib/jvm/java-6-openjdk/jre/bin/java  1061       Автоматически
  1      /usr/lib/jvm/java-6-openjdk/jre/bin/java  1061       Вручную
  2      /usr/lib/jvm/java-6-sun/jre/bin/java      63         Вручную
Нажмите Enter, чтобы подтвердить сделанный выбор [*],
либо укажите номер варианта: 2
update-alternatives: использую /usr/lib/jvm/java-6-sun/jre/bin/java,
чтобы перевести /usr/bin/java (java) на управление вручную.
```

Внутрисистемная информация об управлении ссылками сохраняется в каталоге `/var/lib/alternatives` или `/var/lib/rpm/alternatives` в зависимости от дистрибутива.

13 Библиотеки, Java и Mono

В этой главе мы поговорим о библиотеках, которые необходимы для выполнения программ. Большинство программ для Linux существуют в скомпилированной форме и обращаются к различным библиотекам, которые динамически загружаются по мере необходимости. В разделе 13.1 рассказывается, как в Linux организовано управление библиотеками.

Если вы работаете с одним из распространенных дистрибутивов, то обычно вам приходится устанавливать скомпилированные программы (так называемые двоичные пакеты). Если же вы хотите поработать с совсем новой версией программы или с редко используемым приложением, может получиться так, что вы не найдете заранее скомпилированной версии программы, которую можно просто загрузить. В таких случаях вам придется загрузить исходный код (как правило, он пишется на языках с или C++) и самостоятельно скомпилировать программу. В разделе 13.2 даются некоторые вводные советы о том, как это сделать (не вдаваясь в детали необъятной темы «Разработка программ для Linux»).

В этой главе также рассказано, как выполнять в Linux программы, написанные для Java или .NET. Для этого потребуется установить среду выполнения Java или Mono. Во многих дистрибутивах это делается по умолчанию.

Рассмотрение сценариев, выполняемых интерпретатором, выходит за рамки этой главы. В Linux используются различные сценарные языки, в том числе Perl, Python, PHP (для сайтов) и оболочка bash. В этом издании подробно описана оболочка bash (см. главу 6).

13.1. Библиотеки

Практически все программы Linux используют одни и те же стандартные функции, например для обращения к файлам, вывода изображения на экран, поддержки X и т. д. Было бы нецелесообразно записывать все эти функции прямо в коде не самой большой программы — тогда файлы программ стали бы гигантскими. Вместо этого большинство программ Linux обращается к так называемым *разделяемым библиотекам*: при выполнении программы автоматически загружаются и требуемые библиотеки. В чем заключается преимущество? Если несколько программ используют функции одной и той же библиотеки, то эту библиотеку необходимо загрузить лишь один раз.

Библиотеки играют ключевую роль, когда определяется, какие программы можно будет выполнять на компьютере. Если не хватает одной-единственной библиотеки (или в наличии имеется только старая версия), то непосредственно при запуске программы выводится сообщение об ошибке. Чтобы в таких случаях вы не оставались на произвол судьбы в недрах Linux, в этом разделе я предоставлю базовую информацию по библиотекам.

Динамическая связь программ с помощью ссылок. Большинство программ Linux при работе обращаются к разделяемым библиотекам. Так экономится место на диске (ведь двоичные файлы программ компактны) и меньше нагружается оперативная память (поскольку один и тот же код не требуется грузить многократно). Замечание для программистов, работающих с Windows: разделяемые библиотеки сравнимы с DLL — динамически подключаемыми библиотеками.

Программы, связанные статическими ссылками. При компилировании программ библиотеки можно связывать и статическими ссылками. Это значит, что библиотечные функции интегрируются прямо в программный код. При этом программный файл увеличивается в размерах, зато не зависит от каких-либо библиотек. Это практично, если программу требуется передавать с компьютера на компьютер — тогда она будет работать «с ходу», независимо от того, какие библиотеки уже установлены на компьютере. Иногда статическими ссылками связываются и некоторые простейшие команды для администрирования, чтобы такие команды оставались доступны и тогда, когда разделяемые библиотеки невозможно использовать из-за ошибок в конфигурации.

Форматы и версии библиотек

На протяжении истории Linux в библиотеки не раз вносились изменения, которые были столь же фундаментальны, сколь и несовместимы друг с другом. К числу таких изменений относится, например, замена формата A.OUT на ELF или замена библиотеки `libc` 5 версии на `glibc` версии 2.*n*, причем к последней можно обращаться и как `libc` 6 (в настоящее время актуальна версия `glibc` 2.10).

В обоих случаях замены библиотек были технически оправданны. Новые форматы или версии позволяют с большей легкостью управлять библиотеками и функциями, обеспечивают более полную совместимость различных платформ Linux (Intel, Sun-Sparc, DEC-Alpha) и пр.

Однако при замене возникают проблемы, связанные с тем, что скомпилированные программы могут выполняться только тогда, когда в системе установлены нужные библиотеки и система может их найти. Если вы попытаетесь выполнить программу для `glibc` в старом дистрибутиве, в котором `glibc` не поддерживается, то получите загадочное сообщение об ошибке следующего содержания:

```
root# programmxy
bash: /usr/local/bin/programmxy: No such file or directory
```

Из-за того, что в настоящее время имеются проблемы с поддержкой библиотеки `glibc`, готовящаяся к выходу версия Debian Squeeze предположительно будет использовать не оригинальную библиотеку `glibc`, а полностью совместимую библиотеку `eglibc`. Эта замена не причинит неудобств ни конечным пользователям, ни

разработчикам. Причины, по которым была предпринята такая замена, описаны на странице http://lwn.net/Articles/332000/.

Автоматическая загрузка библиотек

Если вы работаете с Linux только как пользователь, а не как программист, то вы столкнетесь с библиотеками лишь в тот момент, когда какой-то из них будет недоставать. Обычно такие проблемы возникают, если вы постепенно устанавливаете новую программу. При попытке запустить ее выводится сообщение об ошибке, указывающее на отсутствие определенной библиотеки. Часто актуальные версии программ ссылаются на новейшие версии соответствующих библиотек, которые у вас, возможно, еще не установлены. Со старыми программами вероятен прямо противоположный случай. Может быть, они еще ссылаются на устаревшие библиотеки, которые уже не поддерживаются в вашем дистрибутиве.

Определение списка библиотек

Команда `ldd` передается в качестве параметра, который добавляется к полному имени программы. в ответ `ldd` перечисляет все библиотеки, которые нужны для выполнения программы. Кроме того, указывается, где находится подходящая библиотека и какие библиотеки доступны только в устаревшей версии.

```
user$ ldd /bin/cp
   linux-vdso.so.1 => (0x00007fff3bb59000)
   libselinux.so.1 => /lib64/libselinux.so.1 (0x00007f8ab9acc000)
   librt.so.1      => /lib64/librt.so.1 (0x00007f8ab98c4000)
   libacl.so.1     => /lib64/libacl.so.1 (0x00007f8ab96bb000)
   libattr.so.1    => /lib64/libattr.so.1 (0x00007f8ab94b6000)
   libc.so.6       => /lib64/libc.so.6 (0x00007f8ab9127000)
   libdl.so.2      => /lib64/libdl.so.2 (0x00007f8ab8f23000)
   /lib64/ld-linux-x86-64.so.2 (0x00007f8ab9cea000)
   libpthread.so.0 => /lib64/libpthread.so.0 (0x00007f8ab8d06000)
```

Что касается программ X-, KDE- и Gnome, здесь список библиотек гораздо обширнее. Именно по этой причине эти программы запускаются достаточно долго.

Если `ldd` возвратит результат `not a dynamic executable`, то вы имеете дело с программой, которая уже содержит все необходимые библиотеки, то есть это программа со статическими ссылками.

Названия библиотек

Краткая информация о наименованиях библиотек: окончание `.so` указывает, что мы имеем дело с *разделяемой библиотекой*, окончание `.a` определяет *статическую библиотеку*. Цифра указывает номер основной версии. Например, `ls` требует библиотеку `libc` версии 6.

Каталоги, в которых обычно располагаются библиотеки (например, `/lib`, `/usr/lib`, `/usr/local/lib`, `/usr/X11R6/lib` и `/opt/lib`), часто содержат ссылки, связывающие основную версию библиотеки с той, что установлена на вашем компьютере. Так, для `cp` требуется библиотека `ld-linux-x68-64.so.2`. Но на самом деле на компьютере установлена версия `ld-2.14.so`, совместимая «снизу вверх».

```
user$ ls -l /lib/ld*
   /lib/ld-2.14.1.so
   /lib/ld-linux.so.2 -> ld-2.14.1.so
```

Запуск программ

При запуске программ нужно найти и загрузить все библиотеки — за это отвечает так называемый *компоновщик времени выполнения*. При этом учитываются все каталоги, указанные в переменной окружения `LD_LIBRARY_PATH`. Эти каталоги разделяются двоеточиями.

Кроме того, компоновщик интерпретирует файл `/etc/ld.so.cache`. Это двоичный файл, содержащий всю важную информацию о библиотеке (номера версий, пути доступа и т. д.). Он нужен только для того, чтобы сэкономить время, которое компоновщик в противном случае потратил бы на поиск библиотек.

Файл `/etc/ld.so.cache` создается программой `ldconfig`, которая, в свою очередь, интерпретирует `/etc/ld.so.conf`. В этом файле обычно содержится список всех библиотечных каталогов или список ссылок на другие файлы с каталогами (каталоги `/lib` и `/usr/lib` учитываются в любом случае и поэтому отсутствуют в `ld.so.conf` или других конфигурационных файлах. Если кроме `/lib` и `/usr/lib` не придется учитывать никаких каталогов, то `ld.so.conf` можно вообще опустить).

В некоторых дистрибутивах команда `ldconfig` выполняется при каждом запуске компьютера, чтобы гарантировать максимально обновленное состояние файла кэша. Ее всегда нужно выполнять в тех случаях, когда вы вручную устанавливаете новую библиотеку, иначе система не увидит библиотек. Если библиотеки находятся в новом каталоге, нужно соответствующим образом дополнить файл `/etc/ld.so.conf`. При установке пакетов с библиотеками эти задачи обычно выполняет менеджер пакетов.

32- и 64-битные библиотеки

Большинство распространенных дистрибутивов в настоящее время существуют как минимум в двух вариантах сборки: для 32-битных процессоров, совместимых с Intel/AMD, и для 64-битных процессоров, совместимых с Intel/AMD. Разумеется, для 32-битных процессоров предусмотрены только 32-битные библиотеки. Однако, к сожалению, того же нельзя сказать о 64-битных дистрибутивах: были и остаются программы, которые не компилируются для 64-битных систем. Наиболее известная программа такого рода — Acrobat Reader компании Adobe.

/lib и /lib64. Для выполнения 32-битных программ в 64-битных дистрибутивах вам потребуются 32-битные библиотеки. Чтобы можно было избежать конфликтов, библиотеки устанавливаются в различные каталоги. Среди профессионалов в Linux этот метод называется *мультиархитектура*, или *биархитектура*, так как параллельно поддерживается несколько (или две) архитектуры процессоров. В большинстве дистрибутивов встречаются каталоги /lib32 или /lib64, позволяющие не смешивать библиотеки с различной разрядностью. Такая двойственность, разумеется, связана с определенными недостатками: при установке многочисленных библиотек в двух экземплярах тратится больше дискового пространства, кроме того, при этом осложняется техническая поддержка.

В сравнительно старых версиях Debian и Ubuntu для установки в 64-битных системах базы из 32-битных библиотек использовался, как правило, мета-пакет `ia32-libs`.

Multiarch. В настоящее время применяется новый подход, при котором в `/usr/lib` создаются подкаталоги для каждого конкретного варианта архитектуры процессора, например `/usr/lib/x86_64-linux-gnu` (Debian версии 7 и выше, Ubuntu версии 12.04). Достоинство данного подхода заключается в том, что он применим далеко за пределами мира Intel/Pentium, на процессорах с совершенно другими архитектурами.

Идея Multiarch охватывает не только организацию путей каталогов, но и принципы управления пакетами. Например, при необходимости она должна предоставлять возможность устанавливать один и тот же пакет и в 32-битной, и в 64-битной версии. Чтобы четко различать такие варианты, можно указывать для пакета вариант архитектуры с помощью двоеточия, например `gvfs:i386` (означает 32-битную версию пакета `gvfs`). Работа по проверке зависимостей в таком случае удваивается (соответственно, проверяются зависимости для 32-битных и 64-битных версий). Команда `dpkg` уже хорошо поддерживает работу по принципу Multiarch.

Из-за появления концепции Multiarch пакет `ia32-libs` окончательно устаревает. Но он по-прежнему выпускается ради обеспечения совместимости. Но теперь он определяет только зависимость с `ia32-libs-multiarch`. То есть речь идет исключительно о 32-битном пакете. Подробное описание реализации Multiarch в Debian и Ubuntu приводится на следующих сайтах: http://lwn.net/Articles/482952/ и https://lists.ubuntu.com/archives/ubuntu-devel/2011-October/034279.html.

Предварительное связывание

При запуске программы, которая обращается к динамическим библиотекам, нужно установить соединение между программой и библиотеками. Этот процесс именуется *связыванием*. При работе со сложными программами на связывание тратится немало времени.

Программа `prelink` может заранее выяснить информацию, необходимую для связывания. В первый раз этот процесс длится очень долго. При этом требуется просмотреть все исполняемые программы. Файл `/etc/prelink.conf` определяет, какие каталоги с программами и библиотеками учитывает `prelink`. Другие функции можно установить в `/etc/sysconfig/prelink` или `/etc/default/prelink` (Debian, Ubuntu).

В дальнейшем каждая подготовленная таким образом программа будет обращаться к своим библиотекам гораздо быстрее, а значит, и быстрее запускаться. Такое ускорение особенно заметно в OpenOffice или в программах KDE, на запуск которых теперь потребуется вполовину меньше времени, чем раньше. В любом случае данные предварительного связывания необходимо обновлять при каждом обновлении библиотеки.

Предварительное связывание имеет еще один недостаток — эта операция изменяет исполняемые файлы всех программ и библиотек. Кроме того, вы уже не сможете контролировать целостность таких файлов (то есть не сможете убедиться,

что после установки файлы остались прежними). Когда угодно можно отменить любые изменения, внесенные в ходе предварительного связывания, с помощью команды `prelink -ua`. Базовая информация по предварительному связыванию сообщается в справке `man`, а также по следующему адресу: http://www.gentoo.org/doc/en/prelink-howto.xml.

Debian, Ubuntu. Чтобы пользоваться предварительным связыванием в Debian и Ubuntu, необходимо установить пакет `prelink` и задать в файле `/etc/default/prelink` настройку `PRELINKING=yes`. Предварительное связывание будет ежедневно выполняться в качестве одной из задач Cron.

Red Hat, Fedora. Функция предварительного связывания установлена по умолчанию. Данные регулярно обновляются (задача Cron `/etc/cron.daily/prelink`, конфигурационный файл `/etc/sysconfig/prelink`).

SUSE. Чтобы использовать предварительное связывание, необходимо установить пакет `prelink` и указать в файле `/etc/sys-config/prelink` настройку `PRELINKING=yes`. Затем `prelink` будет выполняться модулем YaST после установки любой новой программы или библиотеки (сценарий `/sbin/conf.d/SuSEconfig.prelink`).

13.2. Как самостоятельно компилировать программы

Существует как минимум две причины, по которым вам, возможно, придется компилировать программы для Linux самостоятельно: либо вы не найдете для вашего дистрибутива двоичного пакета с готовой скомпилированной нужной программой, либо вам может понадобиться скомпилировать программу с конфигурацией, отличающейся от стандартной.

Условия. Прежде чем приступить к делу, потребуется выполнить несколько предварительных условий.

- Установить *набор компиляторов проекта GNU* (`gcc` и `gcc-c++`). В них содержатся компиляторы для с и C++.
- Инсталлировать вспомогательные инструменты, в частности `make`, `automake`, `autoconf` и т. д. Эти программы нужны для конфигурации и выполнения компилирования.
- Установить версии различных библиотек, предназначенных для разработки. Названия соответствующих пакетов обычно заканчиваются на `-devel` (Red Hat, SUSE) или `-dev` (Debian, Ubuntu). Например, в `glibc-devel` или `libc6-dev` содержатся инструментальные файлы для базовой библиотеки `glibc`. Понадобятся ли вам еще какие-нибудь инструментальные пакеты, зависит от конкретной программы, которую вы хотите скомпилировать. Если компилятор или компоновщик будет выдавать сообщения об ошибках, гласящие, что для работы не хватает библиотек, это однозначно указывает на то, что вы пропустили какой-то важный инструментальный пакет.

Debian, Ubuntu. Метапакет определяет взаимозависимости между важнейшими инструментальными пакетами. Поэтому при установке `build-essential`

автоматически инсталлируются многие другие пакеты, которые вместе образуют базовую конфигурацию для разработки программ на C/C++.

Fedora. Для выполнения основных предварительных условий лучше всего выполнить команду `yum groupinstall development-tools`. Чтобы разрабатывать программы для KDE и Gnome, также существуют специальные группы пакетов: `kde-software-development` и `gnome-software-development`.

SUSE. Здесь в модуле YaST устанавливаются все пакеты схемы «Базовая среда разработки». Если вы собираетесь писать программы для KDE или Gnome, установите подборки KDE- или GNOME-DEVELOPMENT. Если предпочитаете `zypper`, выполните команду `zypper install -t` *`шаблон`* `devel_basis`, `devel_kde` или `devel_gnome`.

Распаковка кода

Tar. Исходный код обычно лежит в Интернете в виде TAR-архивов (расширения `*.tar.gz`, `*.tgz` или `*.tar.bz2`). После того как скачаете код, распакуйте его в локальный каталог:

```
user$ tar xzf имя.tar.gz    (Для .gz или .tgz)
user$ tar xjf имя.tar.bz2   (Для .bz2)
user$ cd имя
```

Пакеты SRMP. Кроме TAR-архивов, существуют и другие пакеты, содержащие именно тот исходный код, из которого была скомпилирована определенная программа вашего дистрибутива. Такие пакеты с исходным кодом, как правило, находятся на FTP-серверах вашего дистрибутива. Файлы с исходным кодом для тех дистрибутивов, что работают на основе RPM-пакетов, располагаются в пакетах SRMP с расширением `*.src.rpm`. Для установки выполните, как обычно, `rpm -i`:

```
root# rpm -i имя.src.rpm
```

Фактическое место расположения этого кода зависит от конкретного дистрибутива: в Fedora и Red Hat это `/usr/src/redhat/`, а в SUSE это `/usr/src/packages/`.

- `SOURCES/`*`имя`*`.tar.`*`xxx`* содержит сам исходный код. Распаковка TAR-архива производится так, как это описано выше.
- `SOURCES/`*`имя-xxx`*`.patch` (Red Hat) или `SOURCES/`*`имя`*`.dif` (SUSE) содержит дистрибутивно-специфичные изменения исходного кода. Если вы хотите соответствующим образом изменить (пропатчить) файлы кода, выполните следующую команду:

  ```
  user$ cd имя-исходный_каталог
  user$ patch < имя.dif/patch
  ```

  В зависимости от того, какой каталог в данный момент текущий и какая информация о каталоге записана в патч-файле, нужно дополнительно указать параметр `-p1` (см. справку `man patch`).
- `SPECS/`*`имя`*`.spec` содержит описание пакета, также служащее для создания RPM-пакетов. (Если вы хотите сделать RPM-пакет из самостоятельно скомпилированной программы, используйте для этого команду `rpmbuild`, на которой мы более подробно останавливаться не будем. Прочтите `man rpmbuild`!)

Пакеты с исходным кодом для Debian. В дистрибутивах, построенных на основе Debian, исходный код находится в нескольких файлах, которые нужно установить в текущий каталог с помощью команды `apt-get source`.

```
user$ apg-get исходное имя_пакета
```

Теперь в текущем каталоге вы найдете три новых файла и один каталог:

- *имя_пакета*`.dsc` содержит краткое описание пакета;
- *имя_пакета*`.orig.tar.gz` содержит TAR-архив с первоначальным исходным кодом, написанным разработчиком программы;
- *имя_пакета*`.diff.gz` включает в себя все изменения оригинального исходного кода, характерные для Debian или Ubuntu;
- новый каталог *имя_пакета*`/` содержит информацию, уже извлеченную из *имя_пакета*`.diff.gz`, причем все изменения из DIFF-файла уже выполнены.

Компилирование программы

Для компилирования и установки программ нужны три команды, которые иногда называются «три в одном»: `./configure`, `make` и `make install` (далее они будут описаны более подробно). При этом должен быть открыт каталог с исходным кодом.

Сценарий configure. Это сценарий, который проверяет, доступны ли все необходимые программы и библиотеки. Поскольку сценарий находится в локальной папке, его нужно выполнять в виде `./configure`. Этот сценарий адаптирует файл `Makefile`, содержащий все команды, для компилирования и компоновки различных файлов кода. В некоторых (обычно небольших) программах `configure` может отсутствовать. В таком случае сразу выполняйте `make`.

```
user$ ./configure
```

Команда make. Инициирует обработку команд компилирования и компоновки. Теперь вы увидите (иногда практически бесконечные) уведомления и предупреждения о различных процессах компилирования, переполняющие окно консоли. Если не происходит ошибок, можете просто игнорировать эти сообщения. В результате в каталоге с исходным кодом должен появиться исполняемый файл *имя*.

```
user$ make
```

Во многих случаях на этом этапе уже можно запускать программу (команда `./`*имя*) и тестировать ее. Однако обратите внимание, что некоторые службы, в частности сетевые, требуют специальной конфигурации и обычно должны запускаться с помощью сценариев Init-V!

Команда make install. На заключительном этапе мы должны обеспечить всем пользователям доступ к программе. Нам потребуется скопировать файлы программы и, возможно, файлы библиотек в общедоступные каталоги. Для этого необходимы права администратора. Перед выполнением `make install` следует убедиться, что нужная программа еще не установлена! В ином случае ее предварительно потребуется деинсталлировать.

```
root# make install
```

Возможные проблемы

При компилировании программ могут возникать разнообразные проблемы. Чаще всего они связаны с отсутствием каких-либо вспомогательных компилирующих инструментов или библиотек. Как правило, такие проблемы идентифицирует уже `configure` и устранить их совсем не сложно — нужно просто установить недостающий пакет.

Ситуация осложняется, когда `configure` требует библиотеку, которая недоступна в вашем дистрибутиве, либо у вас нет необходимой версии этой библиотеки. Тогда вам придется искать в Интернете нужную библиотеку или, возможно, сначала скомпилировать библиотеку. Что касается сложных программ, например Apache или mplayer, в Интернете имеются точные руководства по компилированию, в которых пошагово описано, что и в какой последовательности необходимо установить и скомпилировать.

Еще хуже, если в процессе компилирования возникает синтаксическая ошибка и процесс компилирования обрывается сообщением об ошибке. Часто причиной этому служит не программная ошибка, а несовместимость вашего компилятора и кода. Некоторые программы можно скомпилировать только определенной версией `gcc` (часто *не* новейшей!), то есть проблема решается установкой нужной версии компилятора. На этот случай в Интернете или в файлах `README`, сопровождающих исходный код, часто можно найти точные указания.

Управление пакетами. Самостоятельно скомпилированные программы и библиотеки вносят путаницу в управление пакетами. Проблема заключается в том, что хотя программа *abc*, скомпилированная вами, уже установлена в системе, база данных RPM или DEB ничего об этом не знает. Если теперь вы попытаетесь установить пакет *xyz*, зависящий от *abc*, то получите сообщение об ошибке, так как, по мнению системы, не выполняются межпакетные взаимозависимости. Однако инсталлировать пакет с помощью `rpm` можно благодаря параметрам `nodeps` и `-force`.

Самое красивое решение — не устанавливать программу с помощью `make install`, а сначала подготовить и установить пакет. Для этого вам нужно познакомиться с командами, отвечающими за упаковку пакетов. В целом, этот метод очень неудобный, особенно если программа должна не один раз тестироваться и компилироваться заново.

Примеры

HelloWorld на языке C. Размеры этой книги не позволяют дать даже вводную информацию о программировании на c и C++. Однако поскольку в прошлом мне постоянно приходилось отвечать на связанные с этим вопросы, я кратко расскажу, как написать и скомпилировать классическую программу Hello World на c и C++. Для C-версии запишем в редакторе следующие строки в файл `hello.c`:

```
// hello.c
#include <stdio.h>
int main(void)
{
  printf("Hello World!\n");
}
```

С помощью следующих команд программа компилируется и выполняется:

```
user$ gcc -o hello hello.c
user$ ./hello
Hello World!
```

HelloWorld на языке C++. Аналогичный код на C++ выглядит так:

```
// hello.cpp
#include <iostream>
int main()
{
  std::cout << "Hello World!\n";
  return 0;
}
```

Теперь для компилирования применяется не gcc, а g++:

```
user$ g++ -o hello hello.cpp
user$ ./hello
Hello World!
```

СОВЕТ

Если вам нужна удобная среда разработки для программирования на с или C++ для Linux, попробуйте KDevelop (KDE) или Anjuta (Gnome). Возможная альтернатива — среда разработки Eclipse для Java, которая, однако, первоначально предназначалась не для с или C++.

Настоящие приверженцы UNIX/Linux считают отличной средой разработки редакторы vi или Emacs.

13.3. Java

На персональных компьютерах Java важен в первую очередь как плагин для браузера, а также для выполнения различных дополнительных функций OpenOffice. Кроме того, значение Java возрастает в силу постоянно увеличивающегося количества программ Java, не зависящих от конкретной платформы. Гораздо более важную роль Java играет на серверах Linux, предлагающих сайты или веб-службы, которые написаны на Java (Tomcat, Jakarta и т. д.). Java интересен и для многих школьников и студентов, которые часто учатся программированию на этом языке, в частности, в среде разработки Eclipse, выполняют проектные работы и т. д.

Стандартная версия Java (Java SE), Java-компилятор javac, части инструментария для разработки на Java (JDK), виртуальная машина Java (JVM) и библиотека классов JDK с 2006 или 2007 года предоставляются в виде открытого кода по лицензии GPL (версия 2).

Варианты Java. Кроме официальных пакетов Java от Oracle, которые предоставляются бесплатно и даже входят в состав некоторых дистрибутивов в форме двоичных пакетов, существуют варианты, состоящие только из открытого кода: они основаны на GPL-коде от Sun, но включают и элементы из других проектов с открытым кодом. Это IcedTea и OpenJDK.

Java на основе OpenJDK, по умолчанию встраиваемый во все большее количество дистрибутивов, демонстрирует практически 99%-ную совместимость с Java от Sun. Оставшийся процент следует списать на те компоненты Java, которые невозможно

предоставить в виде открытого кода из-за проблем с лицензией и для которых еще не написаны свободно распространяемые замены.

Версии и номенклатура Java. Если вам кажется, что нумерация, принятая в ядре Linux, слишком непонятна, это значит, что вы еще не работали с номенклатурой и номерами версий Sun-Java. Только представьте себе: JRE 1.5, Java 2 Standard Edition 1.5 и Java 5 — это одна и та же версия Java. В табл. 13.1 и 13.2 показаны важнейшие сокращения и номера версий, используемые в Java. По-видимому, никто (даже компания Oracle на своем официальном сайте) не приведет в систему названия и номера версий, которые действительно являются официальными.

Таблица 13.1. Сокращения Java

Сокращение	Значение
JVM	Виртуальная машина Java (выполняет программы Java)
JRE	Среда исполнения Java (выполняет программы Java; содержит JVM, а также множество библиотек Java)
JDK	Инструментарий для разработки на Java (для разработки Java-программ)
Java SE = JSE	Стандартная версия Java (для применения на локальном компьютере)
Java EE = JEE	Версия Java для создания распределенных приложений масштаба предприятия (для применения на сервере)
Java ME = JME	Микроверсия Java (применяется на КПК и сотовых телефонах)
JavaFX	Версия Java для веб-приложений (сравнима с Adobe Flash и Microsoft Silverlight)

Таблица 13.2. Номера версий Java SE

Год	Официальное название	Версия JRE/JDK	Внутренний номер версии
1996	Java 1	1.0	1.0
1997	Java 1.1	1.1	1.1
1998	J2SE 1.2 (Java 2)	1.2	1.2
2000	J2SE 1.3 (Java 3)	1.3	1.3
2002	J2SE 1.4 (Java 4)	1.4	1.4
2004	JSE 5 (Java 5)	5	1.5
2006	Java 6 SE	6	1.6
2011	Java SE 7	7	1.7

Выяснение версии установленной Java. Чтобы выяснить версию Java, которая установлена на вашем компьютере, запустите следующую команду. Следующие строки выдадут результат openSUSE 12.1. Если команда `java` недоступна, значит, Java не установлена.

```
user$ java -version
java version "1.6.0_22"
OpenJDK Runtime Environment (IcedTea6 1.10.4) (suse-1.2-x86_64)
OpenJDK 64-Bit Server VM (build 20.0-b11, mixed mode)
```

13.4. Mono

.NET-Framework — это огромная библиотека классов, во многом сходная с библиотеками для Java. Язык C#, разработанный для программирования в .NET, специа-

листы по Java также осваивают без проблем. .NET-Framework и C# вместе образуют объектно-ориентированный фундамент, применяемый в программировании для Windows и веб-программирования, и обойтись без этого фундамента (по крайней мере в мире Microsoft) уже невозможно.

Какое отношение все это имеет к Linux? Хотя сообщество Linux, откровенно говоря, мало связано с Microsoft, вышеописанная концепция пришлась по душе некоторым специалистам, разрабатывающим свободное ПО. Тогда появилось название Mono, объединившее разработки с открытым кодом, использующие язык C# и в значительной мере .NET-Framework. Разработку проекта Mono начала компания Ximian. Позже эту фирму приобрела Novell, и теперь Novell — важнейший спонсор проекта Mono. Кроме того, существует партнерское соглашение между Microsoft и Novell, что позволяет исключить потенциальные проблемы, связанные с патентами (разумеется, только для клиентов Novell).

Тем временем проект Mono хорошо зарекомендовал себя на практике и по умолчанию устанавливается во многих дистрибутивах. Некоторые проекты, связанные с Gnome, базируются на Mono. Таковы, например, Banshee (аудиоплеер), F-Spot (работа с изображениями) и Tomboy (электронная записная книжка), также основаны на Mono.

Совместимость между Mono и .NET-Framework так или иначе не совсем полная, то есть нельзя гарантировать, что вы сможете свободно работать в Linux с любой программой, скомпилированной для Windows. Исчерпывающая информация по проекту Mono и его текущему состоянию (то есть сведения о степени совместимости с C# и .NET-Framework) предлагается на сайте http://www.mono-project.com/.

Проблемы, связанные с патентами, и их решение

Разумеется, проект Mono имеет свои недостатки. Противники указывают на серьезную зависимость проекта от компании Microsoft, которая могла бы попытаться побороться с ним в сфере патентования программного обеспечения. Однако в последнее время стало казаться, что сама Microsoft очень оптимистично настроена по отношению к Mono. Платформа .NET-Framework отличается определенной независимостью и усиливает свои позиции относительно Java. Кроме того, язык C# и .NET-Framework описаны в стандартах Европейской ассоциации производителей компьютеров — как минимум элементы, которые указаны в этих стандартах, должны иметь надежную правовую почву.

Наиболее активное сопротивление Mono долгое время оказывал Red Hat. Ключевое значение имела коллекция патентов, которыми владела компания Open Invention Network, ведь именно с ее учетом было принято решение интегрировать Mono в Fedora. Организация Open Invention Network поддерживается различными фирмами, разрабатывающими свободное ПО, владеет множеством патентов, которые предназначены для защиты от патентных исков, направленных против проектов Linux (в том числе Mono). Даже Debian по умолчанию содержит Mono в своей новой шестой версии Squeeze.

Тем временем, эйфория от Mono снова немного ослабевает. В современных версиях Fedora и Ubuntu Mono по умолчанию отсутствует. Но пакеты с Mono попрежнему есть в наличии, и при необходимости их можно установить.

Внутренняя организация Mono

Mono обычно устанавливается в виде пакета `mono-`*`xxx`*, важнейшая часть которого называется `mono-core`. В ней содержится в том числе компилятор C# mcs, виртуальная машина Mono, собрание .NET-совместимых библиотек (файлы `*.dll` в каталоге `/usr/lib/mono/gac`), а также некоторые конфигурационные файлы Mono (каталог `/etc/mono`).

Программы Mono имеют расширение EXE и предоставляются, подобно программам Java, в виде байт-кода. Чтобы выполнить программу Mono, вы передаете имя EXE-файла команде `mono`. Поскольку на практике этот процесс достаточно сложен, для запуска пользовательских программ Mono существуют маленькие сценарии (посмотрите, например, файл `/usr/bin/f-spot`).

Для Mono-разработчиков создан графический пользовательский интерфейс MonoDevelop, который сначала был построен по образу программы Windows SharpDevelop, но теперь уже не зависит от нее.

14 Администрирование файловой системы

В этой главе описаны различные аспекты администрирования файловой системы. Глава рассчитана на опытных пользователей Linux. В ней рассматриваются следующие темы.

- **Как взаимосвязаны компоненты файловой системы** — в этом разделе подробно описывается, как связаны друг с другом различные фрагменты файловой системы Linux.
- **Имена устройств** — внутри системы Linux жесткие диски и другие носители данных, а также их сегменты запрашиваются через файлы устройств, например `/dev/hda` или `/dev/sdc3`. В этом разделе рассмотрена соответствующая схема нумерации и номенклатуры.
- **Секционирование жесткого диска** — это основная часть установки Linux. Иногда для установки может потребоваться добавить новый сегмент диска. В зависимости от варианта секционирования жесткого диска (MBR или GPT) для изменения секционирования применяются разные инструменты.
- **Типы файловых систем** — сложно найти операционную систему, которая поддерживала бы так же много типов файловой системы, как Linux. В этом разделе обобщены важнейшие варианты использования файловых систем.
- **Управление файловой системой** — в данном разделе рассказано, как можно вручную добавить к файловой системе отдельные сегменты с данными и как автоматизировать этот процесс (`/etc/fstab`).
- **Файловые системы Windows и Linux** — в нескольких разделах, посвященных этой теме, даются советы и указания по работе с файловыми системами ext3, ext4, xfs, vfat и ntfs.
- **CD/DVD** — для CD и DVD с данными существуют собственные типы файловых систем, которые кратко представлены в этом разделе.
- **Внешние носители (USB, FireWire)** — если подключить к компьютеру диск Firewire или USB-флешку, обычно автоматически выводится окно файлового менеджера, через которое можно получить доступ к данным. В этом разделе объясняется, что происходит «за кулисами» и как можно (при необходимости) пользоваться внешними носителями данных вручную.
- **Разделы подкачки** — если в Linux не хватает оперативной памяти для выполнения программ, то части памяти помещаются в так называемые разделы подкачки.

- **RAID** — с помощью RAID (избыточных массивов недорогих/независимых жестких дисков) можно соединять друг с другом разделы нескольких дисков, чтобы таким образом построить более надежную и/или быструю общую систему. В этом разделе кратко рассматриваются основы RAID и описывается, как создать систему RAID-0 (с распределением данных).
- **LVM** — программа управления логическими томами (кратко — LVM) обеспечивает более гибкое управление разделами дисков. Например, она позволяет объединить разделы нескольких жестких дисков в виртуальный диск, изменять размеры разделов «на ходу» и т. д.
- **Зашифрованные файловые системы** — если вы хотите предотвратить доступ посторонних лиц к вашим данным (например, при краже компьютера), нужно зашифровать файлы или файловые системы. Для этого в Linux применяются различные методы; самый популярный из них в настоящее время основан на модуле ядра dm_crypt.
- **SMART** — технология самодиагностики, анализа и отчета, которая позволяет собирать статистические данные при эксплуатации жестких дисков и таким образом заблаговременно распознавать потенциальные проблемы с надежностью системы и не допускать потери информации.
- **SSD-TRIM** — с помощью команды TRIM, поддерживаемой всеми современными твердотельными дисками, можно сообщить операционной системе твердотельного диска (SSD), какие блоки памяти файловой системы окажутся неиспользуемыми после удаления файла. SSD позволяет оптимизировать внутрисистемное использование ячеек памяти.

Разумеется, об администрировании файловой системы можно рассказать гораздо больше, но размеры книги ограничены. Однако необходимо дать несколько дополнительных ссылок.

- **Использование файловой системы** — команды для копирования файлов или создания резервных копий, базовая информация о правах доступа к файлам и т. д. описаны в этой книге в главе 7.
- **Сетевые файловые системы** — Linux позволяет интегрировать в дерево каталогов каталоги с других компьютеров, работающих в той же локальной сети. В этой книге будут специально рассмотрены файловые системы CIFS (Windows/Samba) (см. раздел 20.7) и NFS (UNIX/Linux) (см. главу 21).
- **Дисковые квоты** — здесь речь идет о системе, определяющей, сколько места на диске могут занимать файлы конкретных пользователей. Если заданная граница превышается, то пользователь больше не может сохранять новые файлы. Подробнее об этом можно прочитать по адресу http://www.tldp.org/HOWTO/Quota.html.
- **Кластерные файловые системы** — кластерные, или глобальные, файловые системы объединяют данные нескольких компьютеров в виртуальные системы данных. Так можно создавать гигантские запоминающие устройства для данных и параллельно использовать их содержимое на нескольких компьютерах. В Linux предусмотрено несколько методов создания таких областей памяти, например

OCFS (кластерная файловая система Oracle) или GFS (глобальная файловая система):

- http://oss.oracle.com/projects/ocfs2/;
- http://sources.redhat.com/cluster/gfs/;
- http://ceph.com/.

14.1. Как взаимосвязаны компоненты файловой системы

Порой взаимосвязи, возникающие при построении файловой системы, невероятно запутаны. В этом разделе я попытаюсь кратко и понятно представить важнейшие взаимосвязи. Ради наглядности ограничимся рассмотрением встроенных жестких дисков и обычных файловых систем Linux. CD/DVD-приводы, внешние носители данных, управление логическими томами и системы RAID пока оставим в стороне.

Жесткие диски. Встроенные жесткие диски соотносятся с файлами-устройствами системы Linux. В новых дистрибутивах для всех жестких дисков применяются файлы-устройства вида `/dev/sda`, `/dev/sdb` и т. д. В старых дистрибутивах `/dev/hda`, `/deb/hdb` и т. д. используются для дисков IDE, а `/dev/sda`, `/dev/sdb` — для дисков SATA и SCSI.

Разделы. Чтобы расположить на одном жестком диске несколько независимых друг от друга файловых систем, нужно разбить этот диск на *разделы*. Разделы диска также соотносятся с файлами-устройствами, например `/dev/sda1` соответствует первому разделу первого жесткого диска SATA. Номенклатура файлов-устройств для разделов диска будет подробно рассмотрена позже.

Системный (загрузочный) раздел. При запуске Linux система в первую очередь обращается к системному (корневому) разделу. Номер устройства, или универсальный уникальный идентификатор (UUID) файловой системы, содержащейся в этом разделе, заносится (как часть параметра ядра) в конфигурационный файл GRUB.

Другие разделы. Кроме системного раздела, который требуется обязательно, могут быть и другие разделы, учитываемые уже при запуске Linux. Эти файлы перечислены в файле `/etc/fstab`. Он должен находиться в системном разделе. Данный файл интерпретируется при выполнении процесса Init.

Определение совместимости. При включении разделов диска в дерево каталогов автоматически проверяется совместимость файловых систем. Если компьютер аварийно завершил работу, например из-за перебоев с электричеством, то файловая система автоматически восстанавливается или выполняются другие страховочные мероприятия, призванные предотвратить дальнейшие ошибки совместимости и связанные с ними потери данных. Соответствующий тест совместимости также автоматически запускается по истечении определенного периода использования системы. Детали протекания этого процесса зависят от дистрибутива и конфигурации конкретной системы.

Дерево каталогов вместо букв, обозначающих приводы. В Windows принято запрашивать отдельные файловые системы через буквы, присвоенные различным дисководам (A:, C:, D: и т. д.), в Linux же все файловые системы объединяются в одном дереве каталогов. Доступ к системному разделу осуществляется через корневой каталог /. Стартовые пункты других файловых систем могут варьироваться в зависимости от дистрибутива и конфигурации. Обычно используются подкаталоги `/mnt` или `/media`, например `/media/dvd` для DVD с данными. В новых дистрибутивах для этой цели применяется каталог `/run/media/`*`username`*`/`*`dvdname`*.

Добавление файловых систем. Вы можете, не останавливая работу системы, подключать к дереву каталогов новые файловые системы либо отключать те или иные файловые системы. При подключении внешнего носителя данных (например, USB-флешки) эта операция обычно выполняется автоматически. Если такой «автоматизм» не работает либо этот механизм специально был отключен, то администратор может вручную подсоединять и отключать файловые системы командами `mount` и `unmount` соответственно.

Единственной неизменной частью остается системный раздел: его нельзя отключить от файловой системы в ходе эксплуатации компьютера. Это можно сделать только при завершении работы компьютера.

Типы файловых систем. В Linux поддерживается очень много типов файловых систем. Системный раздел должен относиться к одному из следующих типов систем: ext3, ext4 или xfs. Для остальных разделов выбор еще шире. Можно применить, например, файловые системы Windows, UNIX или Apple.

14.2. Названия устройств для жестких дисков и других носителей данных

В настоящее время связь компьютера с приводами обычно осуществляется в соответствии со стандартами IDE, SATA и SCSI. В табл. 14.1 поясняется значение этих и некоторых других сокращений.

Таблица 14.1. Сокращения

Сокращение	Значение
ATA	Advanced Technology Attachment (Подключение по передовой технологии) — интерфейс для подключения жестких дисков
ATAPI	ATA Packet Interface (Пакетный интерфейс ATA) — расширение ATA для CD- и DVD-приводов
IDE	Integrated Device Electronics (Встроенный интерфейс привода) — альтернативное название PATA
PATA	Parallel ATA (Параллельный интерфейс ATA) — традиционный ATA-интерфейс с параллельной передачей данных
SATA	Serial ATA (Последовательный интерфейс ATA) — новый ATA-интерфейс с последовательной передачей данных
SCSI	Small Computer System Interface (Интерфейс малых компьютерных систем) — альтернатива IDE/SATA

Внутренние свойства ядра

Внутри Linux доступ к встроенным и внешним жестким дискам и их разделам, к приводам CD и DVD, а также к другим носителям данных обеспечивается через файлы-устройства. Это особые файлы, выполняющие роль «интерфейсов» между Linux и аппаратным обеспечением компьютера.

Такие файлы-устройства нужны только для управления системой, то есть, например, для изменения секционирования диска или для подключения определенного раздела к файловой системе. При обычной эксплуатации системы вы имеете доступ ко всей файловой системе через каталоги. При этом символ / означает начало файловой системы. Разрешается подключать новые файловые системы в любой части дерева каталогов, например это может быть дополнительный раздел Linux под названием `/data`, раздел Windows с именем `/media/win`.

В ядре имеется два основных семейства драйверов — для жестких дисков и для других носителей данных.

- **IDE** — драйвер IDE в наше время еще обслуживает старые IDE-диски и приводы IDE-DVD/CD. Код IDE в ядре уже не нов, и, по-видимому, не поддерживается.
- **SCSI** — через систему SCSI управляются не только все SCSI-устройства, но и все приводы, подключенные к шинным системам SATA, USB или Firewire.

  С 2007 года большинство жестких дисков IDE запрашивается через драйвер SCSI. Чтобы обеспечить такое взаимодействие, модуль libata дополняет SCSI-систему ядра функциями PATA. Преимущества очевидны: теперь почти все носители данных обрабатываются одинаково, на основе одного и того же базового кода. Только некоторые старые или экзотические материнские платы либо чипсеты несовместимы с libata и по-прежнему требуют IDE-драйвер.

Названия устройств

Любой жесткий диск, управляемый модулем ядра SCSI (то есть в большинстве компьютеров *все* жесткие диски и Flash-накопители), называется по образцу `/dev/sd`*xy* — `/dev/sda`, `/dev/sdb` и т. д. по порядку. Что касается устройств SATA, в данном случае все устройства по очереди связываются каналами и в качестве названия получают букву. На современных материнских платах обычно предусмотрено минимум 6–8 каналов. Если, например, два жестких диска подсоединяются к SATA-каналам 1 и 3, эти устройства получают имена `/dev/sda` и `/dev/sdb`. Если позже третий жесткий диск подсоединяется к каналу 2, название второго диска изменяется с `/dev/sdb` на `/dev/sdc`.

Если применяется libata, то IDE-устройства также получают названия в порядке `/dev/sda`, `/dev/sdb` и т. д. Обратите внимание на важное отличие по сравнению с обычной номенклатурой IDE — если на одном компьютере и к первому, и ко второму каналам IDE подключено два *ведущих* (master) диска, но к первому IDE-каналу не присоединено *ведомое* устройство (slave), то устройства будут иметь названия `/dev/hda` и `/dev/hdc`. При использовании ядра libata, напротив, те же диски получат названия `/dev/sda` и `/dev/sdb`!

Что касается устройств SCSI, то порядок зависит от ID-номеров этих устройств. Пропуски в последовательности номеров ID не учитываются. Иначе говоря, устройства SCSI получат названия от /dev/sda до /dev/sdc. Как и при работе с устройствами SATA, мы в таком случае можем позже изменить конфигурацию и, следовательно, названия устройств: если будет добавлено четвертое устройство с ID-номером 3, оно получит название /dev/sdc; устройство с номером 5 теперь будет запрашиваться как /dev/sdd. Если одновременно подключаются устройства с шинами различных систем, от BIOS и от используемых PCI-разъемов зависит, какая шинная система будет учитываться в первую очередь.

Внешние USB- и Firewire-устройства обрабатываются как устройства SCSI, причем вместо *x* используется первая свободная буква. При присвоении букв порядок, в котором были подключены устройства, имеет определяющее значение. CD- и DVD-приводы получают собственные названия устройств — в зависимости от дистрибутива это /dev/scdn или /dev/srn (табл. 14.2).

Таблица 14.2. Названия устройств

Устройство	Значение
/dev/sda	Первый жесткий диск
/dev/sdb	Второй жесткий диск
...	
/dev/scd0 или /dev/sr0	Первый CD/DVD-привод
/dev/scd1 или /dev/sr1	Второй CD/DVD-привод
...	

IDE-устройства

Если применяется традиционная IDE-система (например, при работе со старыми компьютерами или дистрибутивами), то жесткие диски IDE запрашиваются через файлы-устройства в форме /dev/hda, /dev/hdb и т. д. Порядок расположения IDE-устройств определяется внутренними кабельными соединениями: /dev/hda обозначает первое устройство на первом IDE-канале; /dev/hdb означает второе устройство на первом IDE-канале; /dev/hdc и /dev/hdd обозначают, соответственно, ведущее и ведомое устройства на втором IDE-канале. Вполне возможно, что два устройства получат названия /dev/hda и /dev/hdc, а /dev/hdb останется неиспользованным — именно в тех случаях, когда и к первому, и ко второму каналам подключены ведущие устройства. Приводы CD-ROM и DVD, присоединенные к шине IDE, обрабатываются как жесткие диски.

Виртуальные носители данных (virtio)

Если Linux работает на виртуальной машине и при этом применяется драйвер virtio, ядро обращается к виртуальным дискам по именам устройств /dev/vda, /dev/vdb и т. д. Драйвер virtio обеспечивает на виртуальной машине особенно эффективную коммуникацию между ядром и системой виртуализации. Системы виртуализации KVM и Xen поддерживают virtio по умолчанию.

Номера разделов (MBR)

Схема нумерации разделов зависит от того, как именно секционирован жесткий диск. В следующем разделе подробно рассказано о том, что в настоящее время применяются два варианта секционирования диска: классический метод MBR, при котором таблица секционирования находится в главной загрузочной записи (Master Boot Record), и новый метод с применением таблиц секционирования GUID (GPT), где аббревиатура GUID означает «глобальный уникальный идентификатор». Второй способ применяется преимущественно при работе с очень крупными жесткими дисками, а также с системами EFI (Extensible Firmware Interface, расширяемый интерфейс для работы с прошивкой).

При нумерации разделов по системе MBR, независимо от того, о чем идет речь (об IDE или SCSI), действует правило, в соответствии с которым цифры от 1 до 4 зарезервированы для основных или дополнительных разделов, а цифры от 5 и далее — для логических разделов в рамках дополнительных. Поэтому достаточно часто в нумерации возникают пробелы. Например, если на жестком диске есть основной, дополнительный и три логических раздела, они будут иметь номера 1, 2, 5, 6 и 7. В табл. 14.3 приведено несколько примеров.

Таблица 14.3. Пример нумерации разделов (MBR)

Устройство	Значение
/dev/sda	Первый диск SCSI/SATA (или первый IDE-диск при использовании ядра libata)
/dev/sda1	Первый основной раздел этого диска
/dev/sdd3	Третий основной раздел четвертого диска SCSI/SATA
/dev/sdd5	Первый логический раздел четвертого диска SCSI/SATA
/dev/sdd15	Одиннадцатый логический раздел четвертого диска SCSI/SATA

Количество разделов, которые могут быть на диске, ограничено: во-первых, исторически сложилось, что на диске может быть максимум четыре основных либо три основных и один дополнительный раздел. Во-вторых, в Linux количество используемых логических разделов ограничено 59 (традиционная IDE-система) или 11 (SCSI/SATA/USB/Firewire/IDE с libata). Таким образом, максимальное общее количество равно 63 или 15.

Номера секционирования (таблицы GPT)

Гораздо проще организуется нумерация разделов жесткого диска при применении таблиц GPT. В таком случае отсутствуют отличия между основными, дополнительными и логическими сегментами. Сегменты диска просто нумеруются по порядку. Но сохраняется ограничение — не более 15 сегментов на одном диске. (Обратите внимание, что в данном случае речь идет о лимите, задаваемом ядром Linux. В одной таблице GPT может присутствовать до 128 сегментов!)

ПРИМЕЧАНИЕ

В некоторых случаях физическая последовательность сегментов может зависеть от нумерации! Предположим, что жесткий диск размером 3 Тбайт разделен на три сегмента по 1 Тбайт (от (/dev/sda1 до /dev/sda3). Впоследствии второй сегмент был удален и в освободившейся области было

создано два новых сегмента, каждый размером по 500 Гбайт. Два этих новых сегмента получат имена устройств /dev/sda2 и /dev/sda4! При MBR-секционировании такой частный случай невозможен, так как между /dev/sda1 и /dev/sda3 при этом можно вставить ровно 1 раздел.

Альтернативные названия устройств

Как уже было сказано, при постепенном добавлении в систему новых жестких дисков SCSI или SATA названия устройств, подключенных ранее, могут изменяться. При замене традиционной системы IDE ядром libata названия устройств также изменяются. Присвоение названий внешним устройствам вообще сложно спрогнозировать. Эти названия зависят от того порядка, в котором устройства подключались.

Чтобы, несмотря на такую вариативность названий устройств, можно было обеспечить стандартный доступ к отдельным устройствам или разделам (например, для сценария резервного копирования), в каталоге `/dev/disk` содержатся дополнительные ссылки на все носители данных, упорядоченные в соответствии с различными критериями.

- `/dev/disk/by-id/`*id* использует ID, состоящие из обозначения шинной системы, названия устройства и номера серии или модели.
- `/dev/disk/by-label/`*label* применяет название, данное файловой системе.
- `/dev/disk/by-path/`*path* использует название пути, состоящее из названия интерфейса PCI, шинной системы и номера раздела. Внимание: если в следующий раз вы подключите USB- или Firewire-устройство к другому USB-разъему, путь изменится!
- `/dev/disk/by-uuid/`*uuid* использует номера UUID файловой системы. UUID — это уникальный ID-номер, присваиваемый файловой системе при форматировании. Он позволяет безошибочно идентифицировать файловые системы, в том числе после изменения конфигурации оборудования.

Количество ссылок в каталогах `/dev/disk` бывает разным. Каталоги `/dev/disk/by-label` и `by-uuid`, например, содержат ссылки только на те разделы, которые имеют названия или UUID. За автоматическое создание ссылок отвечает система udev (см. раздел 7.9). Например, следующая команда `ls` демонстрирует ссылки тестовой системы с жестким диском SATA, USB-флешкой и CF-картой (также подключаемой через USB). Чтобы вывод было удобнее читать, я добавил перед строками небольшие отступы и удалил информацию, касающуюся прав доступа.

```
user$ cd /dev/
user$ ls -lR disk/
disk/by-id:
  scsi-SATA_ST3320620AS_5QF194H9             -> ../../sda
  scsi-SATA_ST3320620AS_5QF194H9-part1       -> ../../sda1
  scsi-SATA_ST3320620AS_5QF194H9-part2       -> ../../sda2
  scsi-SATA_ST3320620AS_5QF194H9-part3       -> ../../sda3
  scsi-SATA_ST3320620AS_5QF194H9-part4       -> ../../sda4
  scsi-SATA_ST3320620AS_5QF194H9-part5       -> ../../sda5
  usb-Generic_USB_CF_Reader_058F312D81B      -> ../../sdc
  ...
```

```
disk/by-path:
  pci-0000:00:1d.7-usb-0:5:1.0-scsi-0:0:0:1   -> ../../sdc
  pci-0000:00:1d.7-usb-0:5:1.0-scsi-0:0:0:3   -> ../../sde
  pci-0000:00:1f.2-scsi-0:0:0:0               -> ../../sda
  pci-0000:00:1f.2-scsi-0:0:0:0-part1         -> ../../sda1
  pci-0000:00:1f.2-scsi-0:0:0:0-part2         -> ../../sda2
  pci-0000:00:1f.2-scsi-0:0:0:0-part3         -> ../../sda3
  pci-0000:00:1f.2-scsi-0:0:0:0-part4         -> ../../sda4
  pci-0000:00:1f.2-scsi-0:0:0:0-part5         -> ../../sda5

disk/by-uuid:
  008f06ef-28be-45c9-acbc-20cda51f712b        -> ../../sda2
  06efc09c-9a3e-4668-81d7-8925c380889e        -> ../../sda5
  366CA8D16CA88D65                            -> ../../sda1
  e35139a5-5871-48fe-9191-df0d003e4ed5        -> ../../sda3
```

14.3. Секционирование жесткого или твердотельного диска

Сам процесс секционирования жесткого или твердотельного диска достаточно тривиален. Однако проблема заключается в том, что основы процесса секционирования были заложены еще в 1980-е годы, когда винчестер размером 50 Мбайт казался *гигантским*, а твердотельный диск никто еще и представить себе не мог. За прошедшие десятилетия технология производства дисковых накопителей сделала колоссальный прорыв, а принципы секционирования очень слабо изменились. По-настоящему современным стандартом хранения данных о секционировании являются, пожалуй, только таблицы GUID, но пока они не слишком распространены (к счастью, эта ситуация стремительно меняется прямо на глазах).

Итак, чтобы не допускать ошибок при секционировании жесткого или твердотельного диска, нужно понимать основы и принципы, заложенные достаточно давно. Поэтому данный раздел получился довольно объемным. (Принципы именования и нумерации разделов в Linux описаны в предыдущем разделе.)

ВНИМАНИЕ

Программы для секционирования могут уничтожить всю информацию на жестком диске! Прежде чем применять их, полностью прочитайте этот раздел! Никогда не изменяйте раздел диска, используемый в данный момент, то есть раздел, файловая система которого подключена к дереву каталогов — mounted.

Инструменты секционирования. При установке практически любого дистрибутива Linux вам предлагаются простые в обслуживании инструменты для секционирования диска. Но лишь в немногих дистрибутивах такие инструменты остаются доступны не только при установке, но и при работе. Например, в SUSE есть специальный YaST-модуль System ▸ Partition (Система ▸ Секционирование). Если же секционирование можно произвести лишь на этапе установки, то вам на выбор предлагается множество специальных инструментов для этого. К важнейшим из них относятся текстовые команды `fdisk` или `parted`, а также графическая программа `gparted`.

LVM. Если вы собираетесь часто вносить изменения в схему разделов диска, вам просто необходимо научиться работать с LVM. LVM добавляет виртуальный слой между физическими разделами жесткого диска и разделами, используемыми для размещения файловых систем, тем самым необычайно упрощая постепенное внесение изменений.

MBR или GPT?

Информацию о секционировании диска можно сохранять двумя способами: в главной загрузочной записи (MBR) или в виде таблицы секционирования GUID (GPT). Метод с применением MBR используется уже не одно десятилетие, но он хорошо подходит только для работы с дисками размером не более 2 Тбайт. Использование GPT является обязательным только в трех случаях:

- при работе с дисками размером 2 Тбайт и более;
- для жестких или твердотельных дисков, которые параллельно применяются и для работы с Linux, и для запуска OS X (то есть, как правило, Mac);
- при необходимости запуска в режиме EFI системы Windows 7 или более новой версии Windows.

Во всех других случаях применение таблиц GPT не является обязательным. Правда, формат GPT входит в состав спецификации EFI, но EFI, разумеется, отлично работает и с дисками, секционированными по системе MBR. Верно и обратное: практически любые компьютеры с обычным BIOS могут запускать Linux, если на диске содержится таблица GPT. (Но при установке GRUB 2 в этом случае нужно зарезервировать отдельный BIOS-GRUB раздел, как это описано в главе 15.)

Решение о выборе системы секционирования принимается один раз и является окончательным. Позже, конечно, эту систему в любой момент можно сменить на другую, но все данные при этом будут потеряны! Я обычно создаю на новых дисках таблицу GPT, так как не люблю возиться с основными, дополнительными и логическими томами. К счастью, беспокоиться о совместимости со старыми версиями Windows мне не приходится.

Инициализация жесткого или твердотельного диска. Современные дистрибутивы Linux приспособлены к работе с носителями данных, использующими как MBR, так и GPT. Но если перед вами стоит задача произвести секционирование с нуля (независимо от того, идет ли речь о совершенно пустом носителе либо о таком, который придется форматировать), набор вариантов в большинстве программ, применяемых при установке, ограничен. На пустых жестких дисках таблица секционирования без вашего участия заносится в главную загрузочную запись, а уже отформатированные жесткие диски при этом не затрагиваются.

Итак, если вы хотите самостоятельно определить тип сегментирования, то придется поработать вручную, и лучше всего воспользоваться при этом «живым диском». В окне терминала необходимо запустить с правами администратора программу `parted` и выполнить соответствующую команду `mklabel` (еще раз обращаю ваше внимание: команда `mklabel xxx` сотрет все данные на вашем жестком диске!):

```
root# parted /dev/sda
(parted) mklabel gpt     (Для GPT)
(parted) mklabel msdos   (Для MBR)
(parted) quit
```

Основные правила

Независимо от того, какой инструмент вы применяете, необходимо учитывать некоторые основные правила.

- Во время работы нельзя вносить изменения в системный раздел. Если вы, например, хотите увеличить его размер, лучше всего запустить компьютер с помощью «живого диска». Для этой цели особенно хороши мини-дистрибутивы, оптимизированные под секционирование жестких дисков, например GParted-Live-CD, Parted Magic и SystemRescueCd:
  - http://gparted.sourceforge.net/livecd.php;
  - http://partedmagic.com/;
  - http://www.sysresccd.org/.
- Можно изменить размер только последнего раздела жесткого диска. Разделы нельзя менять местами.
- Увеличение размера раздела на диске возможно лишь в случае, если после этого раздела есть свободное места. «Передвигать» разделы на жестком диске нельзя.
- Если вы изменяете размер раздела диска, то при этом *не происходит* автоматического изменения размера расположенной на нем файловой системы! Для этого требуются другие команды, различающиеся в зависимости от типа файловой системы.
- На одном жестком диске может быть не более 15 разделов. Если вам этого недостаточно — воспользуйтесь LVM. При использовании MBR один из разделов является расширенным и может принимать в себя другие разделы, но не данные или файловую систему напрямую. Таким образом, максимальное количество разделов уменьшается до 3 основных и 11 логических.
- Если вы изменяете схему разделов жесткого диска, который в данный момент используется (например, потому, что один из разделов этого диска — системный раздел работающей Linux), то программа секционирования дополнительно попросит вас перезапустить компьютер.

  Причина ошибки в том, что ядро Linux не может заново считать таблицу разделов диска при работе системы. Таким образом, изменения сохраняются, но вступают в силу только после перезапуска. Вам *придется* перезапустить Linux, чтобы работать с новой схемой сегментов диска.
- При запуске Windows Vista либо любых других более новых версий Windows на носителях с MBR-секционированием необходимо, чтобы раздел с Windows начинался с сектора 2048. Таким образом, первый мегабайт жесткого диска отводится на главную загрузочную запись или загрузчик операционной системы.

Установочные инструменты всех распространенных дистрибутивов Linux задают для разделов диска размеры, кратные 1 Мбайт. Таким образом, удается избежать проблем при взаимодействии с Microsoft Windows, а также с твердотельными дисками, размер сектора которых равен 4 Кбайт. При проведении секционирования в единицах, превышающих 1 Мбайт, обязательно уделяйте внимание применению низкоуровневых команд секционирования.

Сектора, дорожки, цилиндры и блоки

Исторически сложилось так, что для измерения жестких дисков применяются единицы со следующими названиями.

- **Сектор** (512 байт) — мельчайшей единицей жесткого диска является сектор, состоящий из 512 байт. На некоторых новых жестких дисках размер сектора увеличен и составляет 4 Кбайт, но из соображений совместимости размер сектора и далее объявляется равным 512 байт. Советы по работе с такими жесткими дисками даются далее.
- **Дорожка** (32 256 байт) — на всех современных жестких дисках дорожка состоит из 63 секторов, то есть из 63 · 512 = 32 256 байт. Первоначально по количеству секторов можно было узнать, сколько их на одной окружности жесткого диска. Сегодня это количество является условным. Максимально допустимое значение — 63, так как уже в течение многих лет на хранение количества секторов отводится всего 6 бит.
- **Цилиндр** (8 225 280 байт, около 7,8 Мбайт) — размер цилиндра равен произведению размера дорожки на количество читающих головок. При этом самым большим жестким дискам, используемым в Linux, присваивается условное количество головок, равное 255, таким образом, размер цилиндра оказывается равен 255 · 32 256 = 8 225 280 байт. Количество 255 является максимально возможным, так как для поля предусмотрено всего 8 бит.

  Чтобы узнать количество цилиндров, необходимо разделить мощность жесткого диска на размер цилиндра. Как правило, на жестком диске размером 1 Тбайт имеется 1 000 000 000 000 = 212 байт места, что составляет около 121 600 цилиндров. (Производители жестких дисков охотнее считают в десятичной системе, а не в двоичной, так как мощности кажутся «больше». На самом деле, если считать в двоичной системе, то в одном терабайте будет 240 = 1 099 511 627 776 байт, то есть примерно на 10 % «больше», чем в десятичном терабайте!)

Раньше эти термины отражали физическую архитектуру жесткого диска. Нынешняя архитектура уже давно не такая — даже у традиционных жестких дисков, не говоря уже о SSD. Так или иначе, в некоторых программах для секционирования, в частности в `fdisk`, расчеты по-прежнему проводятся в этих единицах. Разумеется, при работе с Linux границы цилиндров не имеют никакого значения; раздел может начинаться прямо в центре цилиндра (даже условного)!

Что такое блок? Будьте внимательны, когда речь заходит о «блоках». Обычно при этом имеется в виду размер блока, предусмотренный в ядре Linux, — 1024 байт (в частности при работе с командами `df`, `du` и `fdisk`). В файловых системах для

внутренней организации данных предусматриваются значительно более крупные блоки: обычно используется величина в 4096 байт = 4 Кбайт либо кратная ей.

Настройка производительности. По причинам, связанным с производительностью, соответствие границ разделов и цилиндров не всегда оптимально, как минимум в тех случаях, когда условные цилиндры состоят из 63 · 255 секторов, как в Linux. Windows Vista и более новые версии системы Windows работают с цилиндрами, имеющими размер 63 · 240 секторов, то есть размер цилиндра кратен 4 Кбайт. Такой способ расчета дискового пространства на новых дисках с размером сектора 4 Кбайт гарантирует, что каждый блок файловой системы будет соответствовать сектору жесткого диска.

Жесткие и твердотельные диски с размером сектора 4 Кбайт

На новых жестких дисках, а также на твердотельных дисках применяются не привычные сектора по 512 байт, а новые сектора, размер каждого из которых составляет 4096 байт (4 Кбайт). Такая организация дискового пространства имеет преимущества, среди которых — более высокая скорость работы и значительно больший объем жесткого диска. Однако ради обеспечения совместимости жесткие диски с секторами размером по 4 Кбайт воспринимаются в операционной системе как диски с секторами по 512 байт.

Для эффективного использования жестких дисков с секторами по 4 Кбайт следует создавать на них разделы таким образом, чтобы размер начального сектора каждого раздела был кратен 4 Кбайт. Если в вашем случае ситуация иная и файловая система пытается изменить 4-килобайтный диапазон, то при работе жесткому диску необходимо считывать, записывать и модифицировать по 2 сектора размером по 4 Кбайт. Из-за этого процессы, связанные с записью информации на диск, будут сильно тормозиться.

Ранее считалось обычной практикой, что первый раздел диска начинается с сектора 63 (то есть с позиции 63 · 512 байт). При работе со старыми версиями Windows — XP и ниже — это по-прежнему требуется! Но если используется жесткий диск с секторами по 4 Кбайт, то такой устаревший подход не является оптимальным. Поэтому некоторые производители жестких дисков предлагают специальные низкоуровневые инструменты для форматирования. Они позволяют отформатировать жесткий диск таким образом, что внутрисистемно 63 сектор будет начинаться с нового 4-килобайтного сектора. Таким образом, вы сможете достичь в Windows XP оптимальной производительности, но такой подход не работает с более новыми операционными системами. В этой книге я исхожу из того, что диск не форматировался специально для Windows XP.

Новые операционные системы уже работают с учетом новой величины сектора. При установке современные дистрибутивы Linux и Windows задают границы разделов так, чтобы размер раздела был кратным 1 Мбайт. Такой подход в любом случае несовместим с Windows XP, и, следовательно, если вы попробуете установить Windows XP на жестком диске, отформатированном таким образом, могут возникнуть проблемы. Если вы сами создаете разделы и применяете при этом программы,

берущие за единицу секционирования сектора по 512 байт (например, `fdisk` без указания дополнительных параметров), то следите, чтобы границы разделов были кратны 8 секторам.

Техническая информация по оптимальному использованию жестких дисков с секторами по 4 Кбайт приводится на сайте http://lwn.net/Articles/377895/.

Корректировка размера расширенного раздела (только при MBR-секционировании)

Инструменты для секционирования, описанные в этом подразделе, а также вспомогательные инструменты, применяемые при установке Linux, имеют принципиальное различие. Некоторые инструменты оставляют раздел таким же по размеру, каким он был при создании. Эти программы мы отнесем к типу 1. Таковы, например, `fdisk`, `parted`, программы установки Fedora, Red Hat и SUSE, а также программы секционирования для Windows.

Другие программы, напротив, автоматически корректируют размер увеличенного раздела таким образом, чтобы все логические разделы могли разместиться именно в этом физическом разделе (тип 2). Таковы программы установки Debian, Mandriva и Ubuntu.

Сами по себе хороши оба метода, сложности возникают только при их одновременном применении. Например, с помощью инструмента второго типа вы создаете новый логический раздел. При этом программа уменьшает размер расширенного раздела. Если вы теперь попытаетесь создать новый логический раздел с помощью инструмента типа 1, то получите сообщение, что расширенный раздел уже заполнен. Ни одна программа первого типа, за исключением `parted`, не может изменять размеры расширенных разделов, если в них уже находятся логические разделы.

Что делать? Еще раз запустите программу типа 2 и измените существующее разбиение на разделы. Кроме того, с помощью инструмента секционирования второго типа можно создать достаточно большой логический раздел-заполнитель. Таким образом, расширенный раздел автоматически увеличится. Затем вы удалите раздел-заполнитель программой секционирования первого типа и сможете воспользоваться свободным местом в расширенном разделе.

Разумеется, чтобы раз и навсегда избавиться от всей головной боли, связанной с расширенными разделами, можно сразу использовать GPT-секционирование.

Программа fdisk (MBR)

Это одна из самых старых программ Linux, предназначенных для секционирования. Ее пользовательский интерфейс несколько старомоден, однако программа отличается высоким качеством и по достоинству оценена теми, кто не первый год работает с Linux.

ПРИМЕЧАНИЕ

Программа fdisk предназначена только для жестких и твердотельных дисков, на которых применяется MBR-секционирование. Если при запуске fdisk обнаруживает, что диск имеет таблицу GPT, то

эта программа начинает работать по принципу parted. Обратите внимание на этот момент, если не хотите потерять все содержимое вашего диска!

Запуск. Программа `fdisk` всегда может обрабатывать только один жесткий диск — название которого было задано при запуске программы (например, `/dev/sdc` для третьего жесткого диска SATA/SCSI). Если вместо этого сообщить программе параметр `-l`, то `fdisk` выдаст список всех разделов всех жестких дисков. После запуска нажатие клавиши `M` *(menu)* дает краткий обзор команд, которыми можно воспользоваться. Нажатие `P` *(print)* выведет список разделов, которые в настоящий момент имеются на выбранном жестком диске.

Современные версии программы `fdisk` совместимы с новыми жесткими и твердотельными дисками, размер сектора в которых равен 4 Кбайт. `fdisk` автоматически рассчитывает дисковое пространство в секторах и не обеспечивает совместимости с DOS и старыми версиями Windows, до Windows XP включительно. Если такая совместимость вам требуется для технической поддержки старых систем, запускайте `fdisk` с параметрами `-c=dos -u=cylinders`. Но этого ни в коем случае нельзя делать на жестком или твердотельном диске с размером сектора 4 Кбайт! В обратной ситуации, когда приходится обрабатывать новый диск с помощью старой версии `fdisk`, нужно задать при запуске параметры `-c -u`. Так вы не позволите `fdisk` нерационально секционировать разделы.

Создание нового раздела. С помощью клавиши `N` *(new)* создаются новые разделы жесткого диска. При этом можно создать максимум четыре основных раздела. Если требуется управлять более чем четырьмя основными разделами, то один из них должен быть расширенным. Тогда в области, занимаемой расширенным разделом, можно создать до 11 логических разделов. Если создаваемые разделы жесткого диска должны относиться к различным типам (основные, расширенные или логические), команда `fdisk` выдает дополнительный запрос о типе раздела.

Когда будет определено, какого типа должен быть новый раздел, программа спросит, с какого места он должен начинаться (как правило, с первого свободного цилиндра) и какого размера должен быть (то есть в каком цилиндре он заканчивается). Можно указывать размер и более удобным способом — в виде `+`*n*`M` или `+`*n*`G` (то есть `+50G` означает, что размер раздела должен составлять 50 Гбайт).

Определив новый раздел, можно отобразить всю таблицу разбиения на разделы с помощью клавиши `P`. Кроме того, можно определять новые разделы, удалять разделы, определенные ранее, и т. д.

Идентификационные номера разделов. Новые разделы, создаваемые `fdisk`, всегда относятся к типу Linux native (идентификационный номер 83). Если вам требуется раздел другого типа, нужно изменить номер нового раздела, нажав `T` *(type)*. Чаще всего используются разделы следующих типов (идентификационные номера указаны в шестнадцатеричной системе):

- 82 — раздел подкачки Linux;
- 83 — файловая система Linux (любая файловая система Linux — ext, reiser, xfs и т. д.);
- 8e — раздел Linux LVM;
- fd — раздел Linux RAID.

Чтобы получить список всех доступных ID-номеров, нажмите клавишу L. В полученном списке также будут указаны коды для других операционных систем (MS-DOS, Windows, UNIX и т. д.).

Сохранение сделанных изменений. Программа `fdisk` выполняет все изменения только при нажатии клавиши W *(write)*. Перед этим можно проверить нажатием V *(verify)*, верна ли вся внутрисистемная информация о диске. Это делается в качестве перестраховки. Обычно в ответ на такую команду система просто выдает количество секторов по 512 байт, которые не относятся ни к основным, ни к логическим разделам диска (то есть являются неиспользованными).

Если вы запутались, то в любой момент можете завершить работу `fdisk` нажатием клавиши Q *(quit)* или сочетания Ctrl+C — в таком случае жесткий диск останется без изменений.

ПРИМЕЧАНИЕ

Разделы, создаваемые fdisk, еще пусты, то есть в них нет файловой системы! Команды для создания файловой системы зависят от того, какая именно система вам требуется — например, команда mkfs.ext3 создает файловую систему ext3. В разделе 14.13 описано, как вручную создать раздел подкачки.

В табл. 14.4 приведены клавиши для выполнения команд `fdisk`.

Таблица 14.4. Клавиши для выполнения команд fdisk

Клавиша	Значение
D	Удалить раздел (delete)
L	Отобразить идентификационный номер раздела (list)
M	Показать онлайн-справку (menu)
N	Создать новый раздел (new)
P	Отобразить список разделов (print)
Q	Завершить работу программы (не изменяя таблицу разбиения) (quit)
T	Изменить тип раздела (для разделов подкачки)
U	Изменить единицы измерения с секторов на цилиндры и наоборот (unit)
V	Проверить таблицу разбиения (verify)
W	Изменить таблицу разбиения (write)

Увеличение раздела. В принципе `fdisk` не может увеличить размер существующего раздела. Единственное исключение — вы можете изменить размер последнего раздела на жестком диске (или последнего логического раздела в расширенном разделе) при условии, что за этим разделом еще есть свободное место. В таком случае можно удалить данный раздел и создать на его месте новый, большего размера.

Команда `fdisk` изменяет только таблицу разбиения, не затрагивая сами данные, расположенные на диске. Это означает, что файловая система на увеличенном разделе не растет вместе с ним. Таким образом, часть раздела остается неиспользованной. Увеличить файловую систему можно лишь в некоторых дистрибутивах.

Вообще, изменять размер раздела или файловой системы очень опасно, и пусть лучше этим занимаются профессионалы Linux. Если вам необходимо произвести такую операцию, предварительно выполните резервное копирование!

Пример. Рассмотрим, как создать новый раздел на жестком диске, размер сектора которого составляет 4 Кбайт. Клавиша P выводит справку об актуальном состоянии диска. Размер диска составляет около 255 · 63 · 182 401 · 512 байт, что равно примерно 1,36 Тбайт. (Если считать так, как это делают производители жестких дисков, то есть принять, что 1 Тбайт = 1012 байт, то получится 1,5 Тбайт. Но я здесь считаю так, как это происходит в программе fdisk, в степени два. Таким образом, 1 Кбайт = 210 байт, 1 Тбайт = 240 байт.)

Сначала на диске присутствует два основных раздела. Первый раздел начинается с сектора 2048, эта точка расположена на позиции 2048 · 512 байт = 1 Мбайт. Данный раздел состоит из 48 827 392 блоков (размер блока равен 1024 байт). Имеем: 48 827 392 · 1024 байт = около 46,6 Гбайт. Второй раздел начинается с сектора 97 656 832, то есть ровно через 47 684 Мбайт после начала жесткого диска. Размер этого раздела — 1 952 768 · 1024 байт, то есть, около 1,9 Гбайт.

```
root# fdisk -c -u /dev/sda
Команда (m для справки): p
Диск /dev/sda: 1500.3 GByte, 1500301910016 Byte
255 головки, 63 /дорожки, 182401 цилиндра, всего 2930277168 секторов
Единицы = сектора по 1 x 512 = 512 Bytes
Размер сектора (логический/физический): 512 bytes / 512 bytes
Размер ввода-вывода (минимальный/оптимальный): 512 bytes / 512 bytes
Идентификатор диска: 0x0004b057

Загрузка устройства   Начало      Конец       Блоки     Id Система
/dev/sda1    *        2048        97656831    48827392  83 Linux
/dev/sda2             97656832    101562367   1952768   82 Linux Swap / Solaris
```

Клавиша N создает новый, основной раздел. В качестве стартового цилиндра выбирается первый свободный цилиндр. Параметр +30G позволяет не задавать конечный цилиндр, а просто устанавливает размер раздела равным 30 Гбайт.

```
Команда (m для справки): n
Команда Действие
   e     Расширенный
   p     Основной раздел (1-4)
p
Номер раздела (1-4): 3
Первый сектор (101562368-2930277167, Стандартное значение: 101562368): <return>
Использовать стандартное значение 101562368
Последний сектор, +сектора или +sizeK,M,G (101562368-2930277167,
   Стандартное значение: 2930277167): +30G

Команда (m для справки): p
Загрузка устройства.Начало     Конец       Блоки       Id Система
/dev/sda1    *     2048        97656831    48827392    83 Linux
/dev/sda2          97656832    101562367   1952768     82 Linux Swap / Solaris
/dev/sda3          101562368   164476927   31457280    83 Linux
```

Клавиша W сохраняет измененную таблицу секционирования. Поскольку некоторые другие разделы жесткого диска уже используются, нам не удается правильно

считать новую таблицу. То есть перед тем как можно будет использовать новый раздел, компьютер необходимо перезапустить.

```
Команда (m для справки): w
Таблица секционирования разделов была изменена!
Вызовите ioctl(), чтобы заново считать таблицу разбиения разделов.
ПРЕДУПРЕЖДЕНИЕ: Повторное считывание таблицы секционирования аварийно завершилось
с ошибкой 16: Устройство или ресурс заняты. Ядро по-прежнему использует старую таб-
лицу. Новая таблица будет использоваться после перезагрузки компьютера или после
запуска partprobe(8) или kpartx(8).
Синхронизация дисков.
```

Программа parted (MBR и GPT)

К сожалению, работать с этой программой еще сложнее, чем с `fdisk`. Другой недостаток — все сделанные изменения применяются незамедлительно, а не тогда, когда вы прикажете их сохранить. Самое значительное достоинство `parted` по сравнению с `fdisk` заключается в том, что она отлично работает и с таблицами GPT. Подробное описание многочисленных функций `parted` приводится по адресу http://www.gnu.org/software/parted/.

Запуск. При запуске `parted` необходимо указать устройство — жесткий диск. При нажатии клавиши `H` отобразятся команды, предлагаемые на выбор. Ввод `H` *команда* выдает короткий справочный текст по отдельным командам. Нажатие клавиши `P` позволяет вывести таблицу разбиения — в данном случае имеем жесткий диск размером 1,5 Тбайт, полностью занятый тремя RAID-разделами.

```
root# parted /dev/sda
 (parted) print
Модель: ATA WDC WD15EARS-00Z (scsi)
Диск /dev/sda: 1500GB
Размер сектора (логический/физический): 512B/512B
Таблица разбиения: msdos

Количество  Начало Конец  Размер Тип       Файловая система   Флаги
   1        1049kB 50,0GB 50,0GB primary   ext4               boot
   2        50,0GB 52,0GB 2000MB primary   linux-swap(v1)
   3        52,0GB 84,2GB 32,2GB primary
```

По умолчанию `parted` считает в десятичной системе (то есть, 1 Гбайт = 109 байт). Но можно также вести расчеты с применением единиц `KiB`, `MiB`, `GiB` или `TiB`, выражающих, соответственно, килобайты, мегабайты, гигабайты и терабайты в двоичной системе:

```
(parted) unit MiB
(parted) print
...
Диск /dev/sda: 1397GiB
...
Количество  Начало    Конец       Размер    Тип      Файловая система Флаги
   1        1,00MiB   47684MiB    47683MiB  primary  ext4             boot
```

```
2          47684MiB  49591MiB    1907MiB   primary   linux-swap(v1)
3          49591MiB  80311MiB    30720MiB  primary
```

Обслуживание. С помощью команд `mkpart` и `rm` можно, соответственно, создавать и удалять разделы. При работе с `mkpart` необходимо указать желаемые позиции (начальную и конечную) для нового раздела. (К сожалению, мы не можем просто задать желаемый размер раздела и приказать программе вычислить для такого раздела начальную и конечную позиции.) При этом «Гбайт» — сокращенное обозначение гигабайта (109 байт), GiB — сокращенное обозначение гигабайта в двоичной системе (230 байт).

Если вы создаете логический раздел, перед этим необходимо соответствующим образом увеличить расширенный раздел. Команды, изменяющие размеры разделов, ожидают в качестве параметра так называемый *младший* номер устройства, то есть, например, `7` для `/dev/sda7`.

Если вы собираетесь использовать новый раздел в качестве раздела подкачки либо части системы RAID или LVM, то нужно соответствующим образом настроить тип раздела. Необходимая команда выглядит так: `set` *`номер_раздела атрибут`*. В том числе могут применяться `boot`, `swap`, `lvm` и `raid`.

Пример. Следующие команды сначала создают расширенный раздел, а потом логический раздел диска с файловой системой ext4. Расширенный раздел должен примыкать к разделу 3 и занимать все место, оставшееся до конца жесткого диска. (Если задавать отрицательные значения позиций, то они отсчитываются от конца жесткого диска.) Логический раздел должен иметь размер около 40 Гбайт.

```
(parted) mkpart extended 80311MiB -0
ПРЕДУПРЕЖДЕНИЕ: ядру не удалось повторно считать  таблицу секционирования
в /dev/sda (устройство или ресурс занят). В результате после перезагрузки
система может отразить не все внесенные вами изменения.
(parted) mkpart logical 80312MiB 120000MiB
(parted) print
Количество Начало    Конец       Размер     Тип      Файловая система  Флаги
 1         1,00MiB   47684MiB    47683MiB   primary  ext4              boot
 2         47684MiB  49591MiB    1907MiB    primary  linux-swap(v1)
 3         49591MiB  80311MiB    30720MiB   primary
 4         80311MiB  1430799MiB  1350488MiB extended                   lba
 5         80312MiB  120000MiB   39688MiB   logical
 (parted) quit
```

Программа sfdisk (MBR)

По сравнению с `fdisk` и `parted`, команда `sfdisk` достаточно проста. С ее помощью можно построить список разделов жесткого или твердотельного диска и заново секционировать жесткий диск, основываясь на таблице разбиения, предоставляемой в текстовой форме. Применять `sfdisk` удобно прежде всего в тех случаях, когда вы хотите в точности скопировать схему разбиения одного диска на другой, например для конфигурации RAID. Команда `sfdisk` пригодна только для работы с носителями, разбитыми по системе MBR!

Пример. В следующем примере команда `sfdisk-d /dev/sda` узнает список разбиения первого диска, который может прочитать программа `sfdisk`. Символ | передает этот список второй команде `sfdisk`, которая соответствующим образом форматирует второй жесткий диск:

```
root# sfdisk -d /dev/sda | sfdisk /dev/sdb
```

Когда я испытывал эту команду, `sfdisk` иногда жаловалась, что второй жесткий диск уже используется. Убедившись, что это не так (в том числе выполнив `dmesg | grep sdb`), я применил со второй командой `sfdisk` параметр `--force`. Потребовалось дополнительно перезапустить компьютер, чтобы ядро восприняло новое секционирование второго диска:

```
root# sfdisk -d /dev/sda | sfdisk --force /dev/sdb
root# reboot
```

Параметр `--force` следует использовать и в тех случаях, когда `sfdisk` сообщает, что раздел заканчивается не точно на границе цилиндра. В Linux и в большинстве других операционных систем границы цилиндров на жестком диске определяются произвольно и не влияют на работу системы.

Программа gparted (MBR, GPT)

Для `parted` существует графический пользовательский интерфейс `gparted` (рис. 14.1). Эта программа может изменять только те разделы диска, которые в данный момент не используются, то есть не подключены к дереву каталогов. Все применяемые разделы помечаются в программе символом замка, и кнопки для их обработки отключены. В таком случае попробуйте запустить `gparted` с «живого диска» с Linux.

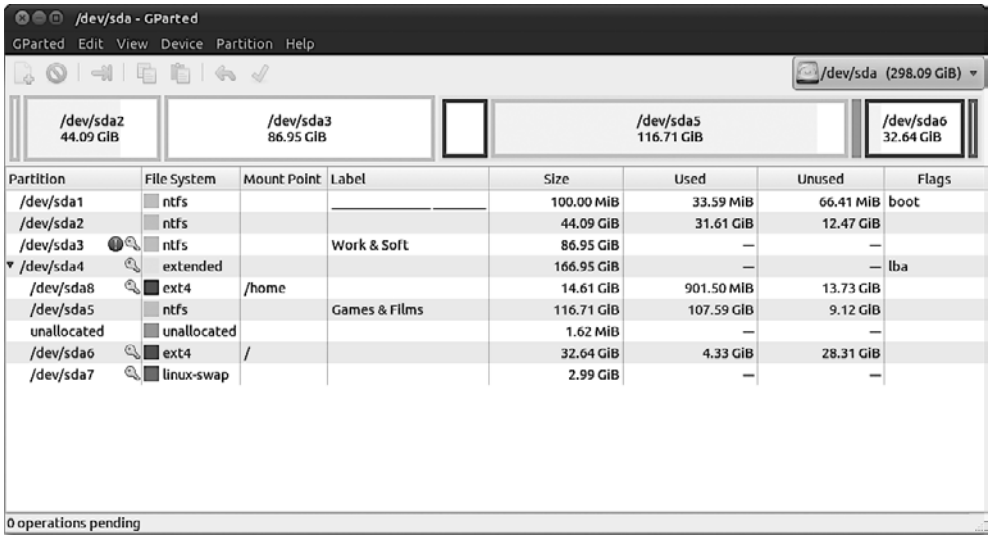


Рис. 14.1. Программа gparted

В отличие от parted, программа gparted запоминает все заданные действия, но сначала не выполняет их. Команда меню Edit ▸ Cancel (Правка ▸ Отменить) отменяет выбранные операции, а Edit ▸ Done (Правка ▸ Принять) наконец выполняет их.

В SUSE в качестве альтернативы gparted можно использовать модуль YaST Система ▸ Разметка диска. В большинстве других дистрибутивов подобные инструменты отсутствуют, что удивительно, так как в установочной программе предусмотрены соответствующие функции.

Gnome Disks (MBR и GPT)

Сравнительно новая программа Gnome Disks помогает при секционировании жестких и твердотельных дисков, при создании файловой системы и при подключении файловых систем к дереву каталогов (в частности, при изменении /etc/fstab). Начиная с версии Gnome 3.6, Gnome Disks заменяет собой более старую программу Palimpsest, выполнявшую подобные задачи в более ранних версиях Gnome. Gnome Disks характеризуется более аккуратным пользовательским интерфейсом (рис. 14.2), но функций в ней меньше, чем в Palimpsest. Gnome Disks находится в пакете gnome-disk-utils, при необходимости ее нужно специально устанавливать.

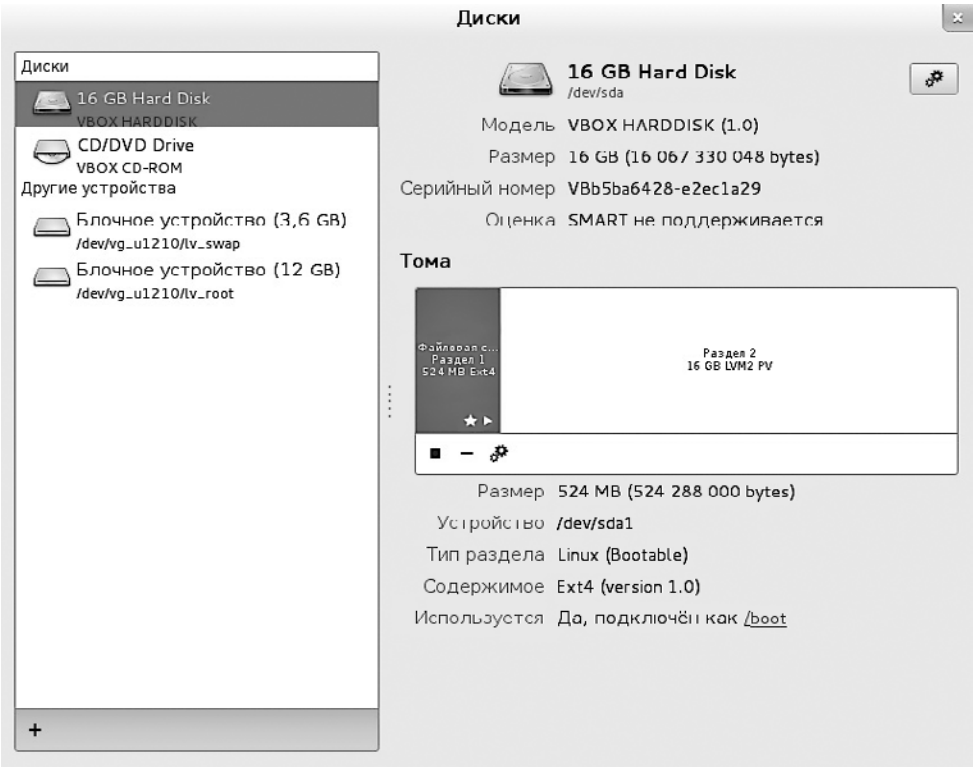


Рис. 14.2. Программа Gnome Disks

В работе с этой программой есть одно досадное обстоятельство. Дело в том, что Gnome Disks требует немедленно настраивать файловую систему на каждом новом сегменте диска. Если вы этого не хотите, выполните команду Тип ▸ Пользовательский и укажите в поле Файловая система заведомо неверное значение (например, `none`). Разумеется, это вызовет сообщение об ошибке, но новый раздел диска будет создан без файловой системы. Далее откройте контекстное меню с помощью кнопки с изображением шестеренки и выберите команду Изменить тип раздела. На данном этапе вы можете отметить раздел для работы с RAID, LVM и т. д.

14.4. Типы файловых систем

В данном разделе мы кратко рассмотрим типы файловых систем, которые могут использоваться в Linux. Наиболее важные файловые системы — ext2–ext4, xfs, vfat, ntfs и iso9660 — описаны далее в этой главе. Чтобы узнать, какой или какие типы файловых систем вы сейчас используете, необходимо выполнить команду `df -T`.

Linux

Файловые системы Linux приспособлены для установки Linux и работы с ней. В повседневной работе вы даже не будете замечать, какую именно файловую систему сейчас используете. Простейшие команды, например `ls` или `cp`, управление правами доступа и др. — все это работает независимо от файловой системы.

Файловые системы отличаются по признакам, представляющим интерес, в первую очередь, для опытных пользователей либо для тех, кто работает с сервером. К этим признакам относятся: скорость обработки внушительных по размеру файлов или большого количества сравнительно небольших файлов, эффективность выполнения операций считывания и записи, нагрузка, оказываемая на процессор, функция журналирования (меры, предпринимаемые после аварийного прекращения работы системы), функции квотирования (возможность ограничить максимальное потребление памяти на пользователя), совместимость с NFS, дополнительные затраты энергии на управление системой, поддержка дополнительных прав доступа (ACL), совместимость с SELinux и т. д.

- **ext** — на начальном этапе развития Linux была доминирующей система ext2 (расширенная файловая система версии 2). С 2002 года ей на смену пришла система ext3, которая во многом совместима с ext2, но к тому же поддерживает функции журналирования, а при работе с версией ядра 2.6 и выше — и ACL. Максимальный размер файла составляет 2 Тбайт, максимальный размер файловой системы — 8 Тбайт. В конце 2008 года было официально заявлено о выпуске версии ext4, которая обладает обратной совместимостью с ext3, но многие функции более эффективны, чем ранее. Кроме того, максимальный размер файловой системы равен 1 Эбайт (1 048 576 Тбайт), и можно рассчитывать, что на какое-то время такого объема хватит.
- **btrfs** — если ситуация сложится так, как пророчат именитые разработчики ядра, то btrfs является для Linux файловой системой будущего. При поддержке Oracle эта файловая система фактически была заново разработана с нуля. В ней преду-

смотрены функции ассоциирования устройств, мгновенные снимки, а также функции RAID. Ближе всего к ней находится система zfs, разработанная Sun. К сожалению, файловая система btrfs еще нуждается в доработке.

- **xfs** — первоначально файловая система xfs разрабатывалась для рабочих станций фирмы SGI, функционировавших в операционной системе IRIX. Xfs особенно хороша для работы с крупными файлами, в частности, идеально подходит для работы с потоковым видео. Система поддерживает квотирование и расширенные атрибуты (ACL).
- **reiser** — система получила название от имени своего основателя, Ганса Райзера (Hans Reiser) и была первой системой с функциями журналирования, обращавшейся за данными к ядру Linux. Версия 3.*n* в SUSE даже считалась стандартной в течение некоторого времени. Основные преимущества reiser по сравнению с ext3 — более высокая скорость работы и эффективность размещения при работе с мелкими файлами. Со временем, правда, разработка reiser приостановилась. Уже давно было заявлено о выходе версии 4, которая все еще не готова, а поддержка версии 3 прекратилась.
- **jfs** — аббревиатура JSF расшифровывается как Journaled File System (Журналируемая файловая система). Первоначально она была разработана для IBM, а затем адаптирована для Linux. Jfs никогда не пользовалась в Linux особым признанием и в настоящее время влачит жалкое существование, уступая другим файловым системам. Подробную информацию о ней вы найдете на сайте http://jfs.sourceforge.net/.

Не существует «быстрейшей» или «наилучшей» файловой системы — оценка зависит от того, для чего вы собираетесь ее применять. Я склоняюсь к использованию ext4 как на настольных, так и на серверных машинах. Система ext3 — тоже неплохой выбор, в первую очередь, в тех случаях, когда важно обеспечить обмен данными с другими операционными системами (то есть, например, если требуется доступ к файловой системе Windows при работе с Linux).

UNIX

Если вы установили на компьютере вторую UNIX-подобную операционную систему, то при обмене данными (передаче из одной ОС в другую) вам пригодятся следующие файловые системы.

- **sysv** — применяется в ОС SCO, Xenix и Coherent.
- **ufs** — используется в FreeBSD, NetBSD, NextStep и SunOS. Linux может только считывать информацию из таких файловых систем, но не может вносить изменения в данные. Для доступа к сегментам с BSD дополнительно потребуется расширение BSD disklabel. Аналогичное расширение существует и для таблиц разбиения SunOS.
- **ZFS** — относительно новая система, разработанная Sun для Solaris. Поскольку код ZFS не соответствует лицензии GPL, ее нельзя интегрировать с ядром Linux. По этой причине Linux поддерживает эту файловую систему лишь опосредованно, через FUSE. Более подробная информация дается здесь: http://zfs-fuse.net.

Windows, Mac OS X

Следующие файловые системы будут полезны при обмене информацией с MS-DOS, Windows, OS/2 и Macintosh.

- **vfat** — используется в Windows 9x/ME. Linux может считывать информацию из таких разделов и вносить в нее изменения. Драйверы системы vfat позволяют работать и со старыми файловыми системами MS-DOS (8 + 3 символа).
- **ntfs** — применяется во всех современных версиях Windows: от NT и выше. Linux может считывать и изменять ее файлы.
- **hfs и hfsplus** — эти файловые системы используются в компьютерах Apple. Linux может считывать и изменять ее файлы — изменять, правда, лишь при условии, что файловые системы были созданы в OS X *без* функций журналирования. На практике такие ситуации, разумеется, являются исключениями — подобная организация целесообразна лишь для обмена данными между OS X и Linux.

CD-ROM/DVD

На CD и DVD с данными обычно используются собственные файловые системы.

- **iso9660** — файловая система для CD-ROM описана в стандарте ISO-9660, допускающем только короткие названия файлов. Длинные названия поддерживаются в различных операционных системах по-разному, с помощью многообразных несовместимых друг с другом расширений. Система Linux способна работать как с расширением Rockridge, обычным в UNIX, так и с расширением Joliet, разработанным Microsoft.
- **udf** — этот формат (универсальный формат диска) появился и развился как наследник ISO 9660.

Сетевые файловые системы

Файловые системы не обязательно должны находиться на локальном диске — они могут подключаться к компьютеру и через сеть. Ядро Linux поддерживает различные сетевые файловые системы, из которых чаще всего применяются следующие.

- **nfs** — важнейшая в UNIX сетевая файловая система (Network File System).
- **smbfs/cifs** — помогают подключать сетевые каталоги Windows или Samba к дереву каталогов.
- **sshfs** — применяется нечасто. Позволяет подключать к локальному дереву каталогов те каталоги, которые доступны по SSH.
- **coda** — эта система очень напоминает NFS. В ней имеется множество дополнительных функций, но она не очень распространена.
- **ncpfs** — работает на базе протокола ядра NetWare, который используется Novell Netware.

Виртуальные файловые системы

В Linux существует несколько файловых систем, предназначенных не для сохранения данных на жестком диске (или другом носителе), а только для обмена информацией между ядром и пользовательскими программами. Эти файловые системы помечены в файле `/proc/filesystems` как `nodev`. Далее коротко рассмотрим важнейшие из них.

- **devpts** — эта файловая система обеспечивает доступ к *псевдотерминалам* (сокращенно — PTY) через `/dev/pts/*` в соответствии со спецификацией UNIX-98. (Псевдотерминалы эмулируют последовательный интерфейс. В системах UNIX/Linux такие интерфейсы используются эмуляторами терминалов, например `xterm`. При этом, как правило, применяются такие устройства, как `/dev/ttypn`. В спецификации UNIX-98, напротив, определяются новые устройства.)
- **proc и sysfs** — файловая система proc служит для отображения служебной информации, касающейся управления ядром и процессами. В дополнение к этому файловая система sysfs строит взаимосвязи между ядром и оборудованием. Обе файловые системы подключаются на позициях `/proc` и `/sys`.
- **tmpfs** — эта система построена на основе разделяемой памяти в соответствии с System V. Она обычно подключается на позиции `/dev/shm` и обеспечивает эффективный обмен информацией между двумя программами.

  В некоторых дистрибутивах каталоги `/var/run` и `/var/lock` также создаются с помощью файловой системы tmpfs. Файлы из этих каталогов применяются некоторыми сетевыми демонами для того, чтобы сохранять идентификационные номера процессов, а также информацию о доступе к файлам. Благодаря tmpfs эти данные теперь отражаются в RAM. Метод гарантирует высокую скорость, а также то, что после отключения компьютера в каталогах `/var/run` или `/var/lock` не останется никаких файлов.
- **usbfs** — файловая система usbfs, начиная с версии ядра 2.6 и выше, дает информацию о подключенных USB-устройствах. Обычно она интегрирована в файловую систему proc.

Прочие файловые системы

В заключение расскажу также о некоторых файловых системах и о ключевых словах, которым не нашлось места в предыдущих подразделах.

- **auto** — на самом деле файловой системы с таким названием не существует. Однако слово `auto` можно использовать в `/etc/fstab` или с командой `mount` для указания файловой системы. В таком случае Linux попытается самостоятельно распознать файловую систему. Этот метод работает с большинством важнейших файловых систем.
- **autofs, autofs4** — это тоже не файловые системы, а расширения ядра, автоматически выполняющие команду `mount` для выбранных файловых систем. Если файловая система не используется в течение некоторого времени, то для нее автоматически выполняется команда `umount`. Этот метод удобен прежде всего

в тех случаях, когда из многих NFS-каталогов одновременно активно используются всего несколько.

При этом во время запуска системы сценарий `/etc/init.d/autofs` автоматически выполняет программу `automount`. Она конфигурируется с помощью файла `/etc/auto.master`. Соответствующие программы автоматически устанавливаются, например, в Red Hat и Fedora. В любом случае autofs активируется только после конфигурации `/etc/auto.master` или `/etc/auto.misc`. Более подробная информация содержится в следующем документе: http://tldp.org/HOWTO/Automount.html.

- **cgroup** — так называемые контрольные группы (Control Groups) позволяют управлять использованием ресурсов с помощью отдельных процессов либо ограничивать таким образом применение ресурсов. Виртуальная файловая система Cgroup помогает считывать или изменять все актуальные настройки Cgroup. Базовая документация о контрольных группах содержится по адресу http://www.kernel.org/doc/Documentation/cgroups/cgroups.txt.
- **cramfs и squashfs** — файловые системы Cram и Squash предназначены только для чтения. Они используются для того, чтобы «упаковать» как можно больше архивированных файлов во Flash-память или ПЗУ (постоянное запоминающее устройство).
- **fuse** — FUSE означает Filesystem in Userspace (Файловая система в пользовательском пространстве). Она позволяет разрабатывать и использовать драйверы файловых систем вне ядра. Следовательно, FUSE всегда применяется с внешним драйвером файловой системы. FUSE работает, в частности, с драйвером NTFS ntfs-3g.
- **gfs и ocfs** — Global File System (Глобальная файловая система) и Oracle Cluster File System (Кластерная файловая система от Oracle) позволяют строить гигантские сетевые файловые системы, к которым могут параллельно обращаться множество компьютеров в один и тот же момент.
- **jffs и yaffs** — Journaling Flash File System (Журналируемая файловая система для Flash-носителей) и Yet Another Flash File System (Альтернативная файловая система для Flash-носителей) оптимизированы для работы с твердотельными дисками и Flash-носителями. С помощью специальных алгоритмов они пытаются равномерно использовать все ячейки памяти (технология «выравнивание износа»), чтобы избежать преждевременного отказа системы.
- **loop** — используется для работы с псевдоустройствами. Псевдоустройство (loopback device) — это адаптер, способный обращаться к обычному файлу как к блочному устройству. Благодаря ему в любом файле можно расположить любую файловую систему, а затем подключить ее к дереву каталогов с помощью `mount`. Отвечающая за это функция ядра — поддержка псевдоустройств — реализуется в модуле loop.

  Существуют разнообразные способы применения псевдоустройств. В частности, они могут использоваться при создании дисков в оперативной памяти для начальной инициализации (Initial RAM disk) для GRUB или LILO, при реализации зашифрованных файловых систем или тестировании ISO-образов для CD.

- **none** — само собой разумеется, что и none не является файловой системой. Эта функция предоставляет редко используемую возможность подключить локальный каталог в другой точке дерева каталогов. Для этого необходимо задать в команде `mount` или в файле `/etc/fstab` тип файловой системы `none`, а также поставить дополнительный параметр `bind`. Этот метод напоминает применение символьной ссылки, но внутри системы реализуется совершенно иначе. Использовать такой метод целесообразно, например, при конфигурации сервера NFS4.
- **unionfs/aufs** — концепция unionfs или ее варианта aufs позволяет как бы накладывать друг на друга несколько файловых систем, причем система, расположенная выше всех, имеет приоритет. Системы unionfs и aufs применяются в некоторых «живых системах», когда Linux запускается прямо с CD или DVD. На систему CD/DVD, предназначенную только для чтения, накладывается файловая система диска оперативной памяти. Снаружи «видна» только одна файловая система, которая составляется из базовой структуры CD/DVD и изменений, производимых в файловой системе диска оперативной памяти.
- **Зашифрованные файловые системы** — в Linux применяются различные методы шифрования содержимого файловой системы. Некоторые из этих методов основаны на применении специальных файловых систем (например, CryptoFS или eCryptfs).

Чтобы узнать, какие файловые системы в настоящее время интегрированы в ядро или загружены в виде модулей, посмотрите файл `/proc/filesystems`. Какие еще модули ядра с дополнительными файловыми системами есть в распоряжении, указано в каталоге `lib/modules/n/kernel/fs/`.

Ссылки

Актуальная информация об имеющихся в распоряжении файловых системах находится в каталоге filesystems документации ядра. Что касается базовой информации, рекомендую почитать файл `HOWTO` о файловых системах. Однако имейте в виду, что последние изменения вносились в эти документы еще летом 2000 года!

- http://www.kernel.org/doc/Documentation/filesystems/;
- http://www.tldp.org/HOWTO/Filesystems-HOWTO.html.

14.5. Управление файловой системой (mount и /etc/fstab)

После установки файловой системы вам, как правило, не придется беспокоиться об управлении файловой системой. Через различные каталоги вы получите доступ если не ко всем разделам жесткого диска, то по крайней мере к большинству из них. Если вы вставите CD или DVD либо подсоедините внешний носитель данных, файловые системы этих устройств автоматически подключатся к дереву каталогов. Все будет работать как по волшебству.

В этом разделе мы заглянем за кулисы системы и подробно рассмотрим команды mount и umount, а также файл /etc/fstab.

- Команды mount и umount всегда выполняются в тех случаях, когда к дереву каталогов подключается (либо отключается от него) раздел диска или носитель с данными. Разумеется, имея права администратора, вы сможете выполнять эти команды и сами, например в тех случаях, когда автоматика отказывает или когда вы работаете без графической системы Рабочего стола (Gnome или KDE).
- Конфигурационный файл /etc/fstab управляет тем, какие файловые системы автоматически подключаются к дереву каталогов при запуске компьютера и какие настройки при этом действуют. Файл /etc/fstab автоматически конфигурируется при установке Linux. Если такая конфигурация вас не устраивает либо требования к системе изменились, этот файл нужно будет изменить в редакторе. В этом разделе я опишу синтаксис данного файла.

Как это ни странно, практически невозможно найти конфигурационный инструмент, который можно было бы использовать вместо команды mount или изменения /etc/fstab вручную. Редкий пример такого инструмента — модуль YaST Система ▸ Разбивка диска (SUSE), а также описанная выше программа Gnome Disks.

Особую группу составляют внешние накопители, например USB-накопители или жесткие диски Firewire. Во многих дистрибутивах такие носители также автоматически интегрируются в файловую систему при подключении к компьютеру. О том, как работать с внешними носителями данных, подробно рассказано в разделе 14.12.

Определение текущего состояния файловой системы

Команда df. Если вам нужно узнать, как в настоящее время организована ваша система Linux, проще всего выполнить команду df. Она указывает, где именно к вашей системе подключены жесткие диски, носители данных и т. д., а также сколько места еще свободно на конкретном жестком диске.

Команда mount. При использовании без дополнительных параметров эта команда сообщает еще более подробную информацию о файловых системах. В итоговом списке будут представлены и различные виртуальные файловые системы. В следующем примере результат выполнения данной команды разбит на колонки, чтобы его было удобнее читать.

```
user$ mount
/dev/mapper/vg-ubuntu on /         type ext4       (rw,errors=remount-ro)
/dev/mapper/vg-myhome on /myhome   type ext4       (rw)
/dev/mapper/vg-virt on /virt       type ext4       (rw)
/dev/sda3 on /boot                 type ext3       (rw)
cgroup    on /sys/fs/cgroup        type tmpfs      (rw,relatime,mode=755)
proc      on /proc                 type proc       (rw)
udev      on /dev                  type devtmpfs   (rw,mode=0755)
```

```
tmpfs     on /run                 type tmpfs     (rw,noexec,nosuid,size=10%,...)
none      on /run/lock            type tmpfs     (rw,noexec,nosuid,nodev,...)
none      on /run/shm             type tmpfs     (rw,nosuid,nodev)
...
```

Информация, напоминающая результат выполнения mount, содержится также в файлах /etc/mtab и /proc/mounts. В каждом из них хранится список всех носителей данных, которые в данный момент подключены к системе, с указанием типа файловой системы и использованных параметров mount. Файл /etc/mtab изменяется всякий раз, когда к дереву каталогов подключается новая файловая система либо отключается одна из файловых систем. Синтаксис mtab такой же, как и у /etc/fstab (см. ниже). В /proc/mounts, кроме того, содержатся параметры, которые прямо не указываются в /etc/fstab или с командой mount.

Как подключать и отключать файловые системы вручную (mount и umount)

После установки современного дистрибутива Linux система конфигурируется так, что команда mount понадобится вам лишь пару раз: все файловые системы Linux будут подключены к дереву каталогов. При вставке CD/DVD или внешнего носителя данных автоматически открывается новое окно файлового менеджера KDE или Gnome. Хотя все и работает, словно по волшебству, система раз за разом выполняет команду mount, чтобы подключать файловые системы к дереву каталогов либо отключать их.

Синтаксис mount выглядит следующим образом:

```
mount [options] device directory
```

Среди параметров указывается в том числе тип файловой системы [-t *xxx*]. Название устройства определяет раздел диска или привод (см. раздел 14.2). В качестве каталога можно указать любой каталог файловой системы, используемой в данный момент. (Он уже должен существовать! При необходимости создайте его с помощью команды mkdir.)

Как правило, команду mount может выполнять только администратор. Однако в /etc/fstab можно разрешить всем пользователям выполнять ее в некоторых разделах диска (параметр user или users).

Примеры. Продемонстрирую на примерах, как работать с mount. В первом примере мы открываем доступ ко всем данным раздела Windows-9x/ME через каталог /windows:

```
root# mkdir /windows
root# mount -t vfat /dev/sda1 /windows
```

Следующая команда подключает привод CD-ROM с диском с данными (файловая система ISO-9660) к общей файловой системе в каталоге /media/cdrom. Устройство /dev/scd0 означает, что привод запрашивается через SCSI-систему ядра. В некоторых дистрибутивах вместо этого нужно указать устройство /dev/sr0.

```
root# mount -t iso9660 /dev/scd0 /media/cdrom
```

Если параметры привода CD-ROM (тип файловой системы, название устройства, каталог) внесены в файл /etc/fstab, то для подключения привода к дереву каталогов достаточно следующей команды:

```
root# mount/media/cdrom
```

Remount. С помощью команды mount -o remount можно изменить настройки уже подключенной файловой системы. Например, следующая команда активизирует параметр exec для DVD — и вы можете выполнять программы, содержащиеся на DVD:

```
root# mount /media/dvd -o remount,exec
```

Если при подключении системного раздела в ходе запуска компьютера возникают проблемы, то раздел подключается в режиме «только для чтения». Однако, чтобы устранить причину ошибки, например исправить запись в файле /etc/fstab, часто требуется вносить изменения в файловую систему. Для этого необходимо выполнить следующую команду.

```
root# mount -o remount,rw /
```

С ее помощью системный раздел подключается заново, и вы можете вносить в него изменения.

Команда unmount. Чтобы отключить файловую систему от дерева каталогов, выполните команду umount:

```
root# umount /media/dvd
```

Автоматическое подключение файловых систем (/etc/fstab)

Было бы очень утомительно заново подключать различные разделы дисков к дереву каталогов при каждом запуске системы и всякий раз, вкладывая CD в привод, выполнять команду mount и указывать различные параметры.

Чтобы облегчить себе работу, используйте файл /etc/fstab: в нем указывается, какие носители данных должны подключаться к файловой системе при запуске компьютера. В любом случае в fstab должна содержаться информация о системном разделе и файловых системах, необходимых для внутрисистемного управления.

Пример. В том или ином дистрибутиве максимально краткий файл fstab может выглядеть так:

```
# Пример двух строк в /etc/fstab
/dev/sda2  /      ext4  defaults   1   1
none       /proc  proc  defaults   0   0
...
```

В первой строке указано, что второй раздел первого жесткого диска используется в качестве системного каталога. В зависимости от того, в каком разделе жесткого диска установлена Linux, вам потребуется указать вместо sda2 имя устройства раздела, содержащего Linux!

Во второй строке к файловой системе подключается система управления процессами. На самом деле файлов и каталогов, расположенных в /proc, на жестком диске нет — мы видим лишь копии файлов, управление оригиналами которых происходит внутри ядра.

Синтаксис /etc/fstab

Из предыдущих примеров уже можно понять, что представляет собой формат fstab: в каждой строке в шести столбцах описывается носитель данных (раздел, файловая система).

Первый столбец. В первом столбце содержится название устройства носителя данных. О номенклатуре разделов жесткого диска рассказано в разделе 14.2. Другие примеры, касающиеся разделов Linux и Windows, приводов CD-ROM и т. д., будут приведены далее в этой главе.

Вместо названия устройства вы можете также указать имя (*имя тома* — в Red Hat и Fedora) или идентификационный номер файловой системы (в Ubuntu). В данном случае действует синтаксис LABEL=string или UUID=*nnn-nnn*. С помощью команды blkid можно узнать название и уникальный идентификатор раздела диска. Для изменения этих данных в различных файловых системах применяются специальные инструменты, например tune2fs.

```
root# blkid /dev/sda9
/dev/sda9: UUID="5a954fc1-00c6-4c25-a943-d4220eff350d" TYPE="ext4"
```

Преимущество этикетки или UUID по сравнению с именованием устройства заключается в том, что данные остаются правильными и в тех случаях, когда название устройства изменяется. Это может особенно легко произойти с USB-носителями. В зависимости от того, какие носители данных применялись раньше, внешний жесткий диск может запрашиваться через /dev/sdc, а в следующий раз — через /dev/sde.

К сожалению, программа fstab достаточно сложна в использовании, особенно при применении уникальных идентификационных номеров. Кроме того, возможны проблемы, если у вас параллельно установлены несколько дистрибутивов Linux. Как правило, при каждой установке нового дистрибутива определенные разделы диска форматируются заново. В таком случае они получают новые уникальные идентификаторы. Дистрибутивы, установленные ранее, перестают узнавать такие разделы, и вам предстоит нелегкая задача — проставить в fstab новые идентификационные номера.

Второй столбец. Во втором столбце определяется, к какому каталогу в дереве файлов подключается носитель данных. Каталоги, указанные во втором столбце, уже должны существовать. Они необязательно должны быть пустыми, но, подключив носитель, вы все равно не будете иметь доступа к файлам, содержащимся в этих каталогах, хотя получите доступ к файлам, которые расположены на подключенном носителе.

Третий столбец. В третьем столбце указывается файловая система. В табл. 14.5 в алфавитном порядке перечислены важнейшие файловые системы. Вы можете указать и несколько файловых систем, разделяя их названия запятыми. Например,

для работы с CD/DVD-приводами можно задать `iso9660,udf`, так как на CD и DVD, как правило, применяются обе эти файловые системы. Команда `mount` автоматически выбирает из указанных файловых систем ту, которая подходит для работы. Обратите внимание, что названия файловых систем нельзя разделять пробелами!

Таблица 14.5. Файловые системы

Файловая система	Использование
auto	Автоматическое определение файловой системы (CD-ROM, дискеты)
brtfs	Файловая система brtfs
cifs	Сетевой каталог Windows (Samba)
devpts	Псевдотерминалы, соответствующие спецификации UNIX-98
ext2, -3, -4	Файловая система ext версий 2, 3 и 4
iso9660	CD-ROM, DVD
nfs	Сетевой каталог UNIX (NFS)
ntfs	Файловая система Windows
proc	Управление процессами (/proc)
reiserfs, reiser4	Файловая система reiser версий 3.n или 4
smbfs	Сетевой каталог Windows (Samba)
swap	Разделы или файлы подкачки
sysfs	Управление системой (/sys)
tmpfs	Обмен данными между программами по системе System V (разделяемая память)
udf	Универсальный формат диска (DVD, CD-RW)
usbfs	Управление USB-устройствами
vfat	Файловая система Windows-9x/ME

Четвертый столбец. В четвертом столбце определяются параметры для доступа к носителям данных. Если задается несколько параметров, они разделяются запятыми. Пробелов быть не должно. В табл. 14.6 перечислены важнейшие универсальные параметры `mount`. Если вы не хотите использовать никаких параметров, применяйте вариант `defaults`.

Таблица 14.6. Основные параметры команды mount

Параметр	Значение
defaults	Использование параметров, заданных по умолчанию
dev	Интерпретация обозначений символьных и блочных устройств
discard	Активизация SSD-Trim (ext4, btrfs, xfs и swap)
exec	Разрешение выполнения программ (например, для CD/DVD-приводов)
noauto	Запрет подключения носителя данных к дереву каталогов при запуске системы
nodev	Игнорирование обозначений символьных и блочных устройств
noexec	Запрет на выполнение программ
nosuid	Запрет интерпретации битов доступа suid и guid
ro	Только для чтения (защита от внесения изменений)
sw	Своп (файл или раздел подкачки)
suid	Интерпретация битов доступа suid и guid

Параметр	Значение
sync	Запрет буферизации доступа, позволяющего вносить изменения (надежнее, но медленнее)
owner	Владелец файла вправе выполнять команды (u)mount
user	Любой пользователь вправе выполнять mount, однако делать это может только тот пользователь, который выполнил mount последним
users	Любой пользователь вправе выполнять команды (u)mount

Пятый столбец. В пятом столбце содержится информация о программе `dump`, и пока этот столбец игнорируется. Здесь принято указывать для системного раздела `1`, а для остальных разделов и носителей данных — `0`.

Шестой столбец. В шестом столбце указано, следует ли проверять файловые системы при запуске компьютера, и если да, то в каком порядке. В большинстве дистрибутивов здесь указывается `1` для системного раздела и `0` — для всех остальных разделов. Это означает, что при запуске системы только системный раздел проверяется на наличие ошибок и при необходимости восстанавливается.

Если вы хотите, чтобы автоматически проверялись и другие разделы, укажите для этих разделов значение `2`. Для всех файловых систем и носителей данных, которые нельзя или не следует проверять, нужно задать `0` (например, для разделов Windows, CD-ROM, DVD, дискет, виртуальных файловых систем, разделов подкачки и др.).

Если в пятом и шестом столбцах `/etc/fstab` нет никаких записей, то система считает, что здесь проставлены нули.

14.6. Основы файловых систем

На следующих страницах речь пойдет в основном о файловых системах ext2, ext3, ext4, btrfs и xfs. Прежде чем описать их создание и администрирование, рассмотрим некоторые базовые данные, не зависящие от конкретной файловой системы.

Журналирование

Все распространенные файловые системы поддерживают функции журналирования. Если описать журналирование предельно просто, то это функция, дополнительно регистрирующая в специальном файле начало и конец каждой операции с файлом. Благодаря этому протоколу позже можно проверить, полностью ли была выполнена та или иная операция с файлом. Если операция не была закончена, ее можно произвести повторно (в области работы с базами данных принят термин *трансакция*, а не *операция*). В самых совершенных системах журналирования также предусмотрена возможность занесения в журнал и конкретных изменений, сделанных в файлах. Это замедляет темп работы, но позволяет более полно реконструировать процесс в будущем.

Теперь, если операцию над файлом не удалось завершить, это видно из протокола. При простом журналировании изменения теряются (то есть чудес эта функция не совершает), однако вы можете просмотреть файл таким, каким он был до внесения изменений.

Основное преимущество функций журналирования заключается в том, что при следующем запуске компьютера система очень быстро возвращается в рабочее состояние и ее можно использовать практически сразу же. Это совсем иная ситуация, нежели ранее, когда после аварийного отключения компьютера или перебоев с электричеством требовалось тщательно проверить файловую систему и исправить возможные ошибки. Этот процесс длился несколько минут, а на особенно больших жестких дисках он мог затянуться на несколько часов.

Потери информации, несмотря на журналирование

При внезапном отключении электричества функция журналирования также не гарантирует полного восстановления файловой системы. Проблема в жестком или твердотельном диске: на нем в целях повышения эффективности при записи информации используется внутренний буфер обмена, поэтому файловая система может получить от диска подтверждение того, что данные получены и сохранены. На самом деле данные еще нужно записать из буфера обмена на жесткий диск, а на это может потребоваться несколько секунд (на твердотельных дисках этот промежуток гораздо короче, но принципиально проблема не решается и на таком диске). Если в этот краткий промежуток времени выскочат пробки, данные из буфера обмена будут потеряны.

Независимо от того, используется ли при записи кэш, поведение жесткого диска после внезапного отключения электричества предугадать сложно. Может быть и так, что жесткий диск запишет не информацию, а случайные биты, до того как пишущая головка внесет информацию на диск. Эта тема обсуждается по следующему адресу: http://lwn.net/Articles/191352/.

Иными словами: система журналирования файлов — вещь хорошая, но не исключает потери данных при неожиданном отключении электричества. Если ваши данные вам дороги, купите UPS (устройство бесперебойного питания), которое даст вам время корректно завершить работу компьютера и при отключении электричества.

Автоматическая проверка файловой системы

Если при запуске Linux обнаружит, что в прошлый раз работа компьютера была завершена некорректно, она проверит файловую систему системного раздела, а также другие разделы, указанные в файле `/etc/fstab` (будет ли проводиться такая проверка, определяется в зависимости от шестого столбца в `/etc/fstab` — см. также подраздел «Синтаксис /etc/fstab» предыдущего раздела). Благодаря применению функций журналирования такая проверка занимает всего несколько секунд.

Кроме того, в некоторых файловых системах (в том числе во всех версиях ext) предусмотрена регулярная проверка системы на предмет ошибок из-за несовместимости. Такие достаточно длительные тесты проводятся при запуске компьютера, если со времени последнего теста истек заданный промежуток времени или было выполнено заданное количество процессов `mount`.

После введения функций журналирования многократно приводились аргументы, что при применении таких функций отпадает необходимость регулярной проверки совместимости. Это справедливо, но, к сожалению, не полностью: файловая система может стать несовместимой и из-за аппаратных ошибок жесткого диска, причем вероятность таких ошибок возрастает по мере увеличения размеров жестких дисков. Например, в техническом паспорте своего жесткого диска (имеющего размер 1 Тбайт) я нашел указание, что вероятность возникновения ошибок в двоичных разрядах *(невосстановимые ошибки чтения для указанного объема прочитанных битов)* не превышает 1 к 1015. Звучит так, как будто этой величиной действительно можно пренебречь. Однако если учесть, что на таком жестком диске умещается $8 \cdot 1012$ бит, становится ясно, что при регулярном использовании этого носителя — без каких-либо повреждений — возникновение ошибок данных вполне вероятно. Если периодически проверять совместимость файловой системы, таких ошибок все равно не предотвратить, зато вы сможете идентифицировать ошибки в работе системы и исправить их (как минимум в тех случаях, когда они затрагивают разделы, критически важные для внутрисистемного управления).

К сожалению, ни одна из файловых систем, описанных в этой книге, не является абсолютно отказоустойчивой. Однако в настоящее время ведутся активные работы по созданию файловых систем, которые могли бы распознавать аппаратные ошибки по неправильным контрольным суммам, а благодаря избыточному хранению данных даже исправлять такие ошибки. Кстати, отличным введением в область исследований, которые сегодня ведутся на ниве создания файловых систем, послужат две следующие статьи:

- http://lwn.net/Articles/190222/;
- http://lwn.net/Articles/196292/.

Вернемся к вопросу проверки файловой системы при запуске компьютера. Детали управления этим процессом часто зависят от конкретного дистрибутива. Как правило, такая проверка проводится без вмешательства пользователя и всего лишь замедляет процесс загрузки. Серьезные неприятности начинаются тогда, когда обнаруживаются неисправимые ошибки. В таком случае раздел диска загружается в режиме «только для чтения». После входа в систему с правами администратора вы можете вручную повторить проверку файловой системы и интерактивно указать, как программа проверки должна поступать с дефектными данными. Как правило, таким образом удается привести систему в состояние совместимости, даже если часть данных спасти не получится.

Проверка файловой системы вручную

Чтобы проверить файловую систему вручную, выполните команду `fsck`. В ходе контроля проверяемый раздел не должен использоваться, то есть предварительно необходимо выполнить `unmount`.

При работе компьютера не получится проверить системный раздел, так как для него нельзя выполнить команду `unmount`. Вместо этого нужно с правами администратора выполнить команду `touch/forcefsck` и перезапустить компьютер. Кроме

того, файл forcefsck создается при выполнении команды shutdown с дополнительным параметром -F.

Если файл /forcefsck существует, то почти во всех дистрибутивах при перезапуске системы автоматически выполняется проверка файловой системы. Если проверка не начнется, загрузите компьютер с помощью восстановительной системы или с «живого диска» (Knoppix) и выполните команду fsck оттуда.

Максимальный размер

Раньше время от времени возникал вопрос о том, каков может быть максимальный размер файла. Ответ зависит от того, какое ядро, процессор с какой архитектурой, какую библиотеку glibc и какую файловую систему вы используете. Все современные дистрибутивы поддерживают расширение библиотеки glibc LFS (LFS означает *«поддержка больших файлов»*). Таким образом, с точки зрения Linux, допустимый размер файла 263 байт представляется практически безграничным. Кроме того, в файловых системах различных типов задаются разные лимиты максимального размера файла (или файловой системы). В табл. 14.7 обобщены соответствующие данные. Помните, что 1 Тбайт = 1024 Гбайт.

Таблица 14.7. Названия устройств разделов жесткого диска

Файловая система	Максимальный размер файла, Тбайт	Максимальный размер файловой системы, Тбайт
brtfs	16 777 216	16 777 216
ext3	2	32 (при величине блока 8 Кбайт)
ext4	16	1 048 576 (или 1 Эбайт)
reiserfs	8	16
reiser4	8	Неизвестно
xfs	9 437 134	9 437 134
ZFS	16 777 216	16 777 216

Обратите внимание, что для создания файловых систем ext4 размером более 16 Тбайт вам потребуется актуальная версия программы e2fsprogs (не ниже версии 1.42).

Изменение типа файловой системы

В большинстве случаев не существует инструментов, которые позволили бы преобразовать файловую систему одного типа в другой (например, из ext3 в reiserfs). Единственный способ — создать нужную файловую систему в новом разделе и скопировать туда все файлы.

14.7. Файловая система ext (ext2, ext3, ext4)

В мире файловых систем Linux доминируют различные версии ext. Проведу короткий экскурс в историю.

- ext, то есть первая версия файловой системы ext, недолго применялась на заре Linux (в 1992 году). Максимальный размер файловой системы составлял 2 Гбайт.
- ext2 была основной файловой системой Linux с 1993 по 2001 год. В этой версии максимальный размер файловой системы составлял 8 Тбайт.
- Важнейшими нововведениями, появившимися в системе ext3, были функции журналирования и поддержка контрольных списков доступа (с версиями ядра 2.6 и выше). Победное шествие ext3 с 2002 года было обусловлено не в последнюю очередь полной совместимостью: имеющиеся системы ext2 не требовалось форматировать заново, с них можно было перейти на ext3 с минимальными усилиями. Если файловая система корректно отключалась командой `unmount`, она даже могла далее использоваться как система ext2.
- В конце 2008 года было официально объявлено, что готова система ext4, и многие современные дистрибутивы уже используют ext4 по умолчанию.

  Важнейшие нововведения: максимальный размер файловой системы достиг 1 Эбайт (1 048 576 Тбайт), время изменения файлов также протоколируется точнее, чем ранее. Так называемые *экстенты* позволяют запрашивать прилегающие друг к другу блоки данных файловой системы как группы, благодаря чему существенно упрощается управление крупными файлами. Кроме того, во многом была оптимизирована скорость работы: и удаление крупных файлов, и проверка файловой системы теперь проводится в разы быстрее, чем в ext3.

  Не остались без внимания и проблемы совместимости: миграция с ext3 на ext4 проходит без малейших проблем. Однако имейте в виду, что при такой миграции обратного пути уже нет!

Совместимость различных версий файловой системы ext выражается и в том, что многие инструменты администрирования до сих пор содержат в имени команды номер версии 2, хотя могут применяться и при работе с новыми версиями (например, `tune2fs`).

Но с момента выпуска ext4 разработка ext, конечно же, не стоит на месте. Правда, в настоящее время еще не планируется выпуск ext5, но в ext4 добавляются новые функции. Не так давно, например, стали рассчитываться контрольные суммы для метаданных. На lwn.net регулярно публикуются новости о состоянии этих разработок, на момент написания книги наиболее актуальной была эта статья: http://lwn.net/Articles/469805.

/etc/fstab. Записи в файле `/etc/fstab`, касающиеся файловых систем ext3 и ext4, обычно выглядят так же, как в следующем примере.

```
# /etc/fstab: Файловые системы Linux
/dev/sdb8   /       ext4    defaults        1  1
/dev/sdb9   /boot   ext3    defaults        0  0
/dev/sdb9   /data   ext4    acl,user_xattr  0  0
```

GRUB. Версия GRUB 0.97 несовместима с ext4! Если хотите загрузиться прямо с системного раздела, содержащего систему ext4, найдите пропатченную версию GRUB 0.97 (поставляется, например, openSUSE до версии 12.1, а также в RHEL 6 но не поставляется с Fedora 11) либо воспользуйтесь GRUB 2. Если в вашем

дистрибутиве применяется оригинальная система GRUB 0.97 без патча для ext4, то вам потребуется отдельный раздел диска с файловой системой в формате ext2 или ext3!

Журналирование

Файловая система ext (версия 3 и выше) поддерживает функции журналирования. Соответствующий файл обычно использует специальные индексные дескрипторы и, следовательно, не виден в файловой системе. В нем содержится информация только о тех файлах, которые были неполностью сохранены на жестком диске. Когда изменения будут выполнены, запись считается *зафиксированной* и может быть заменена новыми записями. Можно (но не принято) записывать файлы журналирования на отдельном устройстве.

В файловой системе ext различаются три метода журналирования.

- `data=ordered` — в этом режиме в журнале сохраняются метаданные, то есть данные о файлах, но не информация, содержащаяся в файлах. В журнале файл обозначается как зафиксированный только тогда, когда он полностью сохранен на жестком диске. После аварийного завершения работы файловая система восстанавливается очень быстро, так как по сведениям журнала можно сразу узнать, какие файлы были сохранены неполностью. Однако такие файлы нельзя восстановить.

  В режиме `data=ordered` журнал каждые 5 секунд синхронизируется с жестким диском. В результате в ext3 все данные по тем или иным файлам физически сохраняются на жестком диске. Такой стандартный метод не очень эффективен, зато очень надежен: даже при общем коллапсе системы или отключении электричества серьезные потери информации практически исключены. В ext3 за `data=ordered` замечен неприятный побочный эффект: при каждом вызове функции `fsync` синхронизируется не только определенный файл, но и вся файловая система. Это может существенно замедлять работу системы.

  В ext4 журнал также синхронизируется с системой каждые 5 секунд, но изменения вносятся в файлы гораздо позже, с использованием технологии отложенного выделения. Чтобы немедленно сохранить файл, необходимо специально вызвать функцию `fsync` (правда, в ext4 для выполнения команды `fsync` не требуется синхронизировать всю файловую систему целиком, поэтому функция выполняется гораздо быстрее).

- `data=writeback` — этот режим напоминает `ordered`. Единственное отличие заключается в том, что работа журнала и операции с файлами не всегда протекают синхронно. Файловая система помечает в журнале данные как фиксированные, не дожидаясь окончательного сохранения информации на диск. В случае аварийного отключения последующая целостность данных гарантирована достаточно надежно. Однако не исключено, что в измененных файлах будут содержаться старые данные. Эта проблема не возникает, если процессы сохранения в пользовательских программах — в соответствии со стандартом POSIX — завершаются командой `fsync`.

- `data=journal` — в отличие от двух предыдущих режимов, при этом в журнале сохраняются не только метаданные, но и сами файлы. Все изменения приходится сохранять дважды (сначала в журнале, а затем в конкретном файле), поэтому система ext3 работает значительно медленнее. Данный метод позволяет восстановить после аварийного отключения и те файлы, изменения которых уже были записаны в журнале, но еще не сохранились в файле.

В принципе информация из журнала переносится на жесткий диск каждые 5 секунд. Этот промежуток можно изменить с помощью параметра `commit` команды `mount`. Если у вас установлен и сконфигурирован пакет `laptop-mode` и ноутбук работает от батареи, то промежуток `commit` будет значительно дольше.

Внутри системы работает демон журналирования `kjournald`, интегрированный в ядро и предназначенный для регулярного обновления файла журнала. Этот процесс запускается автоматически, как только к дереву каталогов командой `mount` подключается файловая система ext3 или ext4.

Отложенное выделение

Важнейшее нововведение системы ext4, влияющее на скорость ее работы, — это так называемое *отложенное выделение* — функция, действующая и во многих других современных файловых системах (например, brtfs, HFS+, reiser4, xfs и ZFS). Отложенное выделение заключается в том, что при внесении изменений в блоки с данными эти блоки резервируются не сразу, а в момент физического сохранения данных на диске — на это может потребоваться до полуминуты. Такой метод имеет два значительных достоинства: во-первых, операции сохранения данных можно выполнять группами, благодаря чему повышается скорость работы и снижается степень фрагментации файловой системы. Во-вторых, временные файлы, которые иногда существуют в течение всего нескольких секунд, зачастую вообще не сохраняются физически.

К сожалению, у отложенного выделения есть и недостатки: основная проблема заключается в том, что метаданные (то есть информация о состоянии файла) часто записываются в систему еще до того, как соответствующие изменения заносятся в файл. При использовании исходного варианта драйвера ext4 случалось так, что измененный, но еще не синхронизированный файл после аварийного отключения системы и последующего восстановления оказывался пуст. Эта проблема особенно часто происходила с конфигурационными файлами. (Многих пользователей устроит, если файл после восстановления системы просто сохранится в исходном состоянии. Однако полная потеря содержимого файла, а вместе с ним и конфигурации программы неприемлема.)

Теодор Цо, главный разработчик всех версий ext, считает, что потери данных обусловлены лишь тем, что во многих программах не выполняется команда `fsync`. Однако в соответствии со стандартом POSIX только выполнение этой команды гарантирует, что изменения действительно будут сохранены. Тем не менее, появились различные изменения драйвера ext4, призванные свести проблему к минимуму: если при изменении имеющихся файлов используются функции `rename` или `ftruncate` (как обычно), то ext4 отказывается от отложенного выделения. Вы можете полностью отключить его, задав параметр `nodelalloc` команды `mount`. Правда, при

этом значительно снижается эффективность работы и часть достижений ext4 в области производительности сводится на нет.

Если вы желаете глубже изучить техническую сторону проблемы, то вас заинтересуют перечисленные ниже сайты. Чтобы их читать, необходимо хорошо знать английский язык и иметь достаточно времени. Однако чтение того стоит: вы подробно изучите, какие концепции лежат в основе современных файловых систем, причем эти сайты будут полезнее иной лекции по информатике!

- http://tytso.livejournal.com/tag/filesystems/ — блог разработчика ext Теодора Цо;
- http://lwn.net/Articles/322823/ — статья о потере данных в ext4;
- http://lwn.net/Articles/326471/ — журналирование в ext3/ext4, fsync;
- http://lwn.net/Articles/327601/ — отчет о работе семинара Linux Storage and Filesystem Workshop.

Стандартная работа системы и дополнительные параметры

Если режим журналирования и распределения данных специально не установить в команде `mount` или файле `/etc/fstab`, то по умолчанию будут действовать следующие настройки:

- ext3 до версии ядра 2.6.29 — `data=ordered`;
- ext3 после версии ядра 2.6.30 — `data=writeback`;
- ext3 после версии ядра 2.6.36 — `data=ordered`;
- ext4 — `data=ordered` с отложенным выделением.

Изменение стандартного режима работы файловой системы ext3 в версии ядра 2.6.30 вызвало немало споров. Еще неизвестно, воспримут ли это изменение все дистрибутивы. Более вероятно, что в отдельных дистрибутивах конфигурация ядра будет изменена так, что ext3 будет работать как раньше, то есть в режиме `data=ordered`. Поэтому, начиная с версии ядра 2.6.36, эта настройка снова активизирована по умолчанию.

Если хотите узнать, какой режим журналирования применяется в настоящее время, прочитайте сообщения ядра. В следующем примере три раздела используют систему ext3, и один раздел — ext4.

```
root# dmesg | grep EXT
EXT3 FS в sda3, внутренний журнал
EXT3-fs: подключенная файловая система, работающая в режиме data = ordered
...
EXT4-fs (sda4): подключенная файловая система, работающая в режиме data = ordered
```

Чтобы специально выбрать определенный режим журналирования, укажите с командой `mount` или в файле `/etc/fstab` параметр `data=`*xxx*. В системе ext4 также можно отключить отложенное выделение параметром `nodelalloc`.

Администрирование

Создание файловой системы. Файловые системы ext2, ext3 и ext4 форматируются с помощью команд `mkfs.ext2`, `mkfs.ext3` и `mkfs.ext4`.

В следующем примере в *логическом томе* размером 20 Гбайт (то есть в разделе диска, управляемом LVM) создается файловая система ext4. Команда `mke2fs` сама задает размер блока данных 4 Кбайт и количество индексных дескрипторов, равное 1 310 720. Это означает, что в данной файловой системе можно расположить максимум 1,3 миллиона файлов. Таким образом, средний размер файла составит 16 Кбайт. Если вы хотите сохранять файлы большего или меньшего размера, можно указать с помощью `-i n`, через сколько байт должен располагаться индексный дескриптор (если средний размер файла будет меньше *n*, то размер файловой системы лимитируется не дисковым пространством, отведенным под раздел, а количеством индексных дескрипторов). Обратите внимание, что изменить абсолютное количество индексных дескрипторов вы уже не сможете. В большинстве случаев вам подойдет значение, задаваемое `mkfs.ext4` по умолчанию.

```
root# mkfs.ext4 /dev/mapper/vg1-test
mke2fs 1.41.4 (12-Jun-2012)
Обозначение файловой системы=
Тип ОС: Linux
Размер блока=4096 (log=2)
Размер фрагмента=4096 (log=2)
1310720 индексных дескрипторов, 5242880 блоков
262144 блоков (5.00%) зарезервировано для суперпользователя
Внешний блок данных=0
Максимальные блоки файловой системы=4294967296
160 групп блоков
32768 блоков в группе, 32768 фрагментов в группе
8192 индексных дескрипторов в группе
Резервные копии суперблоков, сохраненные в блоках:
    32768, 98304, 163840, 229376, 294912, 819200, 884736, 1605632, 2654208,
    4096000
Запись таблиц индексных дескрипторов: готово
Создание журнала (32768 блоков): готово
Запись суперблоков и учетной информации файловой системы: готово
```

Преобразование ext3 в ext4. Если хотите преобразовать имеющуюся файловую систему ext3 в ext4, просто выполните команду, указанную ниже. Команду `tune2fs` можно выполнять «на ходу»; но чтобы использовать новые функции ext4, файловую систему необходимо заново подключить к дереву каталогов. Команда `mount -o remount` не работает.

```
root# tune2fs -O extents /dev/sda5
root# umount /dev/sda5
root# mount /dev/sda5 /home
```

При постепенном преобразовании файловой системы ext3 есть недостаток — имеющиеся файлы не будут использовать экстенты (а новые — будут). В данном случае вам поможет программа дефрагментации `e4defrag`, воспользоваться которой пока нельзя.

Проверка файловой системы. Файловые системы ext регулярно проверяются на наличие ошибок при запуске компьютера, а именно после выполнения определенного количества операций `mount` (по умолчанию 36) либо по истечении

определенного времени (6 месяцев) в зависимости от того, какое из условий будет выполнено раньше. Обратите внимание — в некоторых дистрибутивах максимальное количество операций mount, после которых запускается проверка, либо временной интервал имеет большее значение или это значение равно 0 (проверка не выполняется). Кроме того, во многих дистрибутивах конфигурация fstab такова (если она вообще задается), что проверяется только системный раздел (это касается шестого столбца в fstab — см. подраздел «Синтаксис /etc/fstab» раздела 14.5).

Несмотря на применение функций журналирования, проверять файловую систему рекомендуется как минимум один-два раза в год. Во-первых, так заблаговременно распознаются ошибки оборудования. Во-вторых, не исключено, что в драйверах файловой системы могут быть еще не известные ошибки. Чем раньше будут распознаны ошибки, которые могут возникнуть в результате, тем меньше будет потенциальный ущерб.

Чтобы проверить файловую систему вручную, можно просто выполнить команду fsck.ext2/ext3/ext4. Во время контроля проверяемый раздел не может использоваться, то есть при необходимости сначала выполните unmount.

```
root# fsck.ext4 -f /dev/mapper/vg1-test
...
/dev/mapper/vg1-test: 21357/1310720 файлов (1.3% без взаимосвязей),
    2062135/5242880 блоков
```

Обычно после проверки оказывается, что все в порядке. В противном случае в каталоге /lost+found каждого раздела сохраняются остатки тех файлов, которые уже невозможно восстановить. Если это текстовые файлы, то из их остатков вы, возможно, сможете извлечь крупицы полезной информации.

Настройка интервала для автоматической проверки системы. Действующие интервалы для автоматической проверки файловой системы можно определить и изменить с помощью команды tune2fs. Здесь с использованием параметра -c вы указываете максимальное количество операций mount, а с помощью параметра -i — временной интервал в днях:

```
root# tune2fs -l /dev/mapper/vg1-test
...
Mount count:              1
Maximum mount count:      35
Last checked:             Wed Jul 15 11:45:54 2009
Check interval:           15552000 (6 months)
...
root# tune2fs -c 100 -i 90 /dev/mapper/vg1-test
Установка максимального количества подключений 100
Установка интервала между проверками 7776000 секунд
```

С помощью команды tune2fs -c 0 -i 0 вы полностью деактивируете автоматическую проверку файловой системы.

Настройка названий разделов. С помощью команды e2label можно узнать, настроить или изменить внутреннее название файловой системы ext3 *(имя тома файловой системы)*:

```
root# e2label /dev/sda1 mylabel
```

Это имя можно задать в первом столбце файла /etc/fstab вместо названия устройства.

Настройка уникального идентификационного номера. При создании файловая система автоматически получает уникальный идентификатор (UUID), узнать который можно, выполнив команду blkid. При необходимости эти номера можно изменить с помощью команды tune2fs -U. Такое изменение можно внести на ходу, использовать команду umount не требуется.

```
root# tune2fs -U random /dev/sda1     (Случайные UUID)
root# tune2fs -U f7c49568-8955-4ffa-9f52-9b2ba9877021 /dev/sda1
                                      (Собственные UUID)
```

Изменение размера файловой системы. Команда resize2fs позволяет увеличить или уменьшить размер файловой системы ext. Обратите внимание, что при увеличении системы нужно сначала увеличить раздел или логический том, в котором она находится, а при уменьшении файловой системы нужно сначала уменьшить файловую систему, а потом — раздел диска или логический том. В следующем примере мы увеличим логический том с помощью команды lvextend.

```
root# lvextend -L 40G /dev/mapper/vg1-test
  Расширение тестирования логического тома до 40,00 Гбайт
  Тестирование логического тома выполнено, изменение размера
root# resize2fs /dev/mapper/vg1-test
resize2fs 1.41.4 (27-Jan-2009)
Файловая система на /dev/mapper/vg1-test подключена к /test;
  Необходимо изменить размер онлайн
old desc_blocks = 2, new_desc_blocks = 3
Выполнение изменения размера онлайн /dev/mapper/vg1-test
  на 10485760 (4k) блоков.
Файловая система на /dev/mapper/vg1-test теперь имеет размер 10485760 блоков.
```

Размер файловой системы можно увеличить, не выключая компьютер. Правда, в различных источниках в Интернете пишут, что увеличить систему, которая уже имеет размер 16 Тбайт, очень сложно или вообще невозможно (я не смог протестировать эту возможность, так как у меня не было достаточно крупного массива RAID). Чтобы уменьшить размер файловой системы, ее необходимо отключить от дерева каталогов.

Фрагментация файловой системы. Под *фрагментацией* понимается состояние диска, при котором файлы сохраняются не в последовательно идущих друг за другом блоках, а в произвольном порядке, по всему разделу. Такое состояние может возникнуть, если попеременно удалять, сохранять, удлинять и укорачивать файлы. Фрагментация может значительно замедлить операции доступа к данным.

Драйверы ext2/3/4 пытаются, насколько это возможно, избежать фрагментации. Это, однако, удается сделать лишь в том случае, когда файловая система заполнена не более чем на 90 %.

Пока не существует инструментов, которые позволили бы дефрагментировать файловые системы ext. В настоящее время подобный инструмент для файловой системы ext4 находится в разработке. Эта программа — если она все же будет готова — позволит проводить дефрагментацию диска на ходу.

Доступ к файловым системам ext2 и ext3 из Windows. Вы можете обращаться к данным, сохраненным в Linux, и из системы Windows. Актуальный список программ, которые можно для этого использовать, а также перечень соответствующих драйверов находится по адресу http://wiki.ubuntuusers.de/Linux-Partitionen_unter_Windows.

14.8. Файловая система btrfs

По всей видимости, btrfs становится той файловой системой Linux, которой принадлежит будущее. Несомненно, btrfs — самая современная из файловых систем Linux, существующих на данный момент. Сначала разработка системы btrfs была инициирована Oracle, но теперь в ней участвуют также многие другие фирмы и разработчики ядра. Драйвер btrfs интегрирован в ядро и (как и весь код ядра) распространяется по лицензии GPL.

Функции. Далее перечислены важнейшие свойства btrfs:

- копирование при записи: измененные блоки данных не затираются, а пересохраняются в другом месте. Это совместно с функциями журналирования обеспечивает особую надежность при изменении файлов;
- автоматический расчет контрольных сумм, позволяющий обнаруживать битовые (разрядные) ошибки;
- непосредственная поддержка RAID-0, 1 и 10;
- мгновенные снимки и подтома;
- сжатие файлов (`mount`-параметр `compress`);
- оптимизация работы с твердотельными дисками (`mount`-параметр `ssd`);
- дефрагментация без приостановления работы.

В планах разработчиков также реализация еще некоторых функций, но по состоянию на лето 2012 года они пока отсутствуют. Эти функции таковы: поддержка для RAID-5 и 6, перепроверка файловой системы без приостановки работы, а также функция дедупликации, позволяющая однократно сохранять любые избыточные данные (например, два одинаковых файла).

Подтома (subvolumen), мгновенные снимки (snapshot) и RAID (избыточные массивы независимых дисков) — это функции btrfs, у которых уже есть аналоги в ядре (речь идет о функциях ядра Multi Device (поддержка множественных устройств) и Logical Volume Manager (Менеджер логических томов) — см. разделы 14.14 и 14.15). Сами по себе такие случаи функционального дублирования в ядре нежелательны, но в случае с `btrfs` они были сочтены уместными. Почему? с одной стороны, при непосредственной интеграции функций RAID в драйвер файловой системы на основании контрольных сумм удается повысить надежность данных. С другой — разработчики `btrfs` смогли убедительно доказать, что мгновенные снимки `btrfs` гораздо надежнее подобных снимков, создаваемых с помощью LVM.

Статус. Учтите, что система `btrfs` по-прежнему сыровата и нуждается в доработке, пусть даже специалисты из Oracle и SUSE утверждают обратное и рекомен-

дуют устанавливать btrfs в дистрибутивах для крупных предприятий! В почтовых рассылках btrfs регулярно встречаются сообщения о потерях крупных объемов данных. Вы совершенно не застрахованы от того, что команда mount неожиданно выдаст сообщение об ошибке в только что безупречно функционировавшей системе и работать с этой системой станет невозможно. Перед тем как настраивать btrfs на клиентском или серверном компьютере, обязательно прочтите архив рассылки btrfs за последние два-три месяца (http://dir.gmane.org/gmane.comp.file-systems.btrfs)!

Правда, уже с весны 2012 года применяется команда btrfsck, предназначенная для исправления дефектных файловых систем. Но этот инструмент также пока недоработан и недостаточно тщательно протестирован. Обобщенное описание других известных проблем приводится на следующих сайтах:

- https://btrfs.wiki.kernel.org/index.php/FAQ;
- https://btrfs.wiki.kernel.org/index.php/Gotchas;
- https://btrfs.wiki.kernel.org/index.php/Problem_FAQ.

Несмотря на эти недостатки, большинство дистрибутивов в настоящее время поддерживает btrfs уже во время установки. Но ни в одном дистрибутиве пока нет достаточно удобной программы с графическим интерфейсом, которая упрощала бы работу по администрированию.

Контрольные точки. По сравнению с ext4, btrfs предлагает массу дополнительных функций, но лишь иногда превосходит ext4 по скорости. В принципе применение эталонных тестов с использованием контрольных точек для оценки скорости работы файловой системы — сложный и довольно спорный метод. Тем не менее, раньше btrfs достаточно редко выходила победителем из соревнований с другими системами по скорости работы. Итак, если для вас прежде всего важна скорость — то btrfs явно не лучший вариант. Ссылки на проведенные недавно эталонные тесты скорости работы вы найдете по следующему адресу: https://btrfs.wiki.kernel.org/index.php/Main_Page.

GRUB. GRUB 0.97 несовместим с btrfs. GRUB 2, в основном, поддерживает btrfs начиная с версии 1.99, но не работает с некоторыми вариантами и особыми сборками btrfs. Чтобы справиться с подобными проблемами, вы должны при использовании GRUB 2 самостоятельно применять собственный загрузочный раздел ext*n*.

Администрирование

Администрирование btrfs происходит, как правило, с помощью команды btrfs. Эта команда находится в одном пакете с mkfs.brtfs и brtfsck, и данный пакет в зависимости от дистрибутива может называться btrfstools, btrfs-progrs или btrfsprogs. Если вы не настроили файловую систему brtfs уже при установке дистрибутива, то этот пакет, как правило, приходится устанавливать отдельно.

Настройка файловой системы brtfs. Чтобы настроить новую файловую систему btrfs в пустом сегменте диска или в пустом логическом томе, выполните следующую команду (в данном случае, естественно, нужно заменить /dev/sdb1 собственным именем устройства):

```
root# mkfs.btrfs /dev/sdb1
fs created label (null) on /dev/sdb1
      nodesize 4096 leafsize 4096 sectorsize 4096 size 20.00GB
Btrfs Btrfs v0.19
```

Затем подключаем файловую систему к дереву каталогов:

```
root# mkdir /media/btrfs
root# mount /dev/sdb1 /media/btrfs
```

Увеличение и уменьшение файловой системы. Если файловая система `btrfs` кажется слишком маленькой, то для ее увеличения проще всего добавить к ней еще одно устройство (то есть еще один сегмент диска или логическое устройство). Но вы также можете изменять размер имеющейся системы `btrfs`, не останавливая работу (можно не только увеличивать, но и уменьшать ее!). На практике это удается лучше всего, если файловая система находится в логическом томе.

Для увеличения таким образом файловой системы `btrfs` (чтобы она полностью заняла логический том, предварительно увеличенный с помощью `lvextend`) выполните следующую команду:

```
root# btrfs filesystem resize max /media/btrfs
```

Вместо `max` здесь можно задать новый абсолютный размер файловой системы либо сделать относительное изменение ее размера с помощью `+` или `-`. При этом можно использовать сокращения `k`, `m` и `g`. Они означают, соответственно, килобайт, мегабайт и гигабайт. Следующая команда уменьшает размер файловой системы на 2 Гбайт:

```
root# btrfs filesystem resize -2g /media/btrfs
```

Проверка файловой системы. Чтобы проверить корректность файловой системы, в `btrfs` предусмотрена команда `btrfsck`. Эта команда может выполняться лишь в такой файловой системе, которая еще не подключена к дереву каталогов. В качестве единственного параметра вы задаете имя устройства — это будет имя того сегмента диска, в котором находится файловая система `btrfs`.

```
root# btrfsck /dev/sdb1
```

Преобразование ext3/ext4 в btrfs. Команда `btrfs-convert` позволяет преобразовать имеющуюся файловую систему ext3 или ext4 в `btrfs`. Преобразование протекает удивительно быстро, так как при этом заново закладываются лишь метаданные `btrfs`, а сами блоки данных остаются без изменений. В следующем примере предполагается, что исходная файловая система ext4 находится в каталоге `/dev/sdb1`.

```
root# fsck.ext4 -f /dev/sdb1
root# btrfs-convert /dev/sdb1
Создание метаданных btrfs.
Создание файла образа ext2fs.
Очистка системного раздела.
Преобразование завершено.
root# mount /dev/sdb1 /media/btrfs
```

При преобразовании файловой системы команда btrfs-convert создает мгновенный снимок ext2_saved, архивирующий состояние системы ext до преобразований. Пока у вас есть этот мгновенный снимок, вы даже можете преобразовать систему btrfs обратно в ext! (Но изменения, которые, возможно, будут внесены в систему btrfs между преобразованиями «туда и обратно», в таком случае будут потеряны.)

```
root# umount /dev/sdb1
root# btrfs-convert -r /dev/sdb1
```

Если после преобразования будет нужно продолжить работу именно с btrfs, то нужно удалить мгновенный снимок ext2_saved. Чем больше изменений вносится в файловую систему btrfs, тем больше места занимает мгновенный снимок.

```
root# btrfs subvolume delete /media/btrfs/ext2_saved
```

Базовая техническая информация о таком процессе преобразования изложена на сайте https://btrfs.wiki.kernel.org/index.php/Conversion_from_Ext3.

Сжатие файлов. btrfs поддерживает автоматическое и при этом прозрачное сжатие файлов. Для обеспечения сжатия файловая система должна подключаться к дереву каталогов с mount-параметром compress=zlib или compress=lzo. Параметр compress действует лишь для новых или измененных файлов. Имеющиеся файлы остаются без изменений, если они просто считываются. Этот параметр действует для всей файловой системы, то есть его нельзя применить только к отдельным каталогам. Для сжатия предлагается на выбор либо алгоритм zlib, обеспечивающий более качественный процесс сжатия, либо алгоритм lzo, который работает быстрее, но менее экономно расходует дисковое пространство. (Подробнее см. в статье http://lwn.net/Articles/411577/). Ведутся даже дискуссии о том, что в перспективе lzo-сжатие может быть активировано по умолчанию.

Во многих случаях compress ускоряет операции с файлами. На первый взгляд это может показаться удивительным, так как сжатие и последующее возвращение к исходному размеру вызывает дополнительные издержки при работе с файлами. Если процессор очень быстрый, эти издержки оказываются совсем незначительными по сравнению с сэкономленным временем. Время экономится за счет того, что приходится считывать или изменять меньшее количество блоков с данными на диске. В качестве дополнительной выгоды следует упомянуть, конечно, и экономию места.

Разумеется, впоследствии вы можете использовать сжатую файловую систему и без параметра compress. В таком случае сжатие новых или измененных файлов происходить не будет, но уже сжатые файлы останутся сжатыми, пока такие файлы только считываются.

Параметр compress особенно хорош при работе с каталогами, в которых содержится большое количество текстовых файлов (например, /usr/; при сжатии его размер можно сократить почти на 50 %). Но рекомендуется использовать этот параметр только в вашем домашнем каталоге и при условии, что там, преимущественно, находятся именно сжатые файлы (например, аудио- и видеофайлы, а также файлы в формате PDF и OpenOffice). Дальнейшее сжатие файлов после этого является невозможным. Такая невозможность опознается и самим драйвером btrfs,

который в таком случае не производит дальнейшего сжатия файла. Поэтому такой тест длится довольно долго.

На практике при работе с настольными системами целесообразно применять compress к системному разделу, но с домашним разделом этот параметр лучше не использовать. К сожалению, не в каждом дистрибутиве можно настраивать параметры mount уже при установке. В openSUSE это как раз возможно, и при тестовой установке с compress=zlib удалось уменьшить размер стандартного системного раздела с 3,5 до 1,8 Гбайт!

Конечно, бывают и особые случаи. Если, например, вы используете сервер базы данных MySQL и применяете при этом табличный драйвер InnoDB, автоматически выполняющий сжатие таблиц (это не касается наиболее современных версий MySQL), то каталог с базой данных MySQL (как правило, это /var/lib/mysql) не должен находиться в файловой системе btrfs с параметром compress.

Подтома

Из опыта работы с традиционными файловыми системами вы, вероятно, знаете правило «один раздел (один логический том) — одна файловая система». С btrfs ситуация обстоит иначе: с помощью подтомов (subvolumen) можно в определенном объеме настроить в рамках одной файловой системы btrfs несколько других виртуальных файловых систем и использовать их в собственном mount-каталоге.

Создание подтомов. Проще всего изучить создание подтомов на примере. При этом я исхожу из того, что файловая система btrfs находится у вас в разделе /dev/sdb1 и подключена в каталоге /media/btrfs.

Команда btrfs subvolume create позволяет создать два новых подтома: sub1 и data/sub2. Команда mount с параметром subvol=name подключает подтома к дереву каталогов.

```
root# btrfs subvolume create /media/btrfs/sub1
root# btrfs subvolume create /media/btrfs/data/sub2
root# mkdir /media/sub{1,2}
root# mount -o subvol=sub1 /dev/sdb1 /media/sub1
```

Команда mount -o subvol=name работает только с такими подтомами, которые находятся непосредственно в корневом каталоге файловой системы btrfs. Если подтом расположен где-то в другом месте, то необходимо узнать идентификационный номер этого подтома с помощью btrfs subvolume list, а потом задать этот номер с помощью mount-параметра subvolid:

```
root# btrfs subvolume list /media/btrfs
ID 257 top level 5 path sub1
ID 258 top level 5 path data/sub2
root# mount -o subvolid=258 /dev/sdb1 /media/sub2
```

Теперь можно использовать /media/sub1 и /media/sub2 как две отдельные файловые системы. Все находящиеся там файлы можно найти и непосредственно в файловой системе btrfs, в каталогах sub1 и data/sub2, которые автоматически создаются после выполнения команды btrfs subvolume create.

```
root# touch /media/sub1/tst1
root# ls /media/btrfs/sub1
tst1
root# touch /media/sub2/tst2
root# ls /media/btrfs/data/sub2
tst2
```

Иными словами: подтома можно применять как самостоятельные файловые системы, но читать и изменять их можно только в каталогах, относящихся к файловой системе btrfs.

Подтом, задаваемый по умолчанию. С помощью команды btrfs subvolume set-default можно задать подтом, который будет по умолчанию использоваться при выполнении следующей команды mount, если явно не указать другой подтом в параметрах subvol или subvolid команды mount. Команде set-default необходимо передать ID тома, который вы предварительно узнали с помощью btrfs subvolume list. Если вы выполните следующие команды, то в каталоге /media/btrfs будут находиться файлы подтома sub1:

```
root# btrfs subvolume set-default 257 /media/btrfs/
root# umount /media/btrfs
root# mount /dev/sdb1 /media/btrfs    (Подтом sub1)
```

Как при необходимости снова активировать корневой каталог btrfs? Внутрисистемно этот каталог обрабатывается как подтом, имеющий идентификационный номер 5. (Если вы внимательно изучите вывод команды btrfs subvolume-list, то обнаружите, что 5 задается как идентификационный номер верхнего уровня.)

```
root# btrfs subvolume set-default 5 /media/btrfs/
root# umount /media/btrfs
root# mount /dev/sdb1 /media/btrfs    (Верхний уровень)
```

Удаление подтомов. Команда btrfs subvolume delete name, где name — имя тома, позволяет удалить определенный подтом вместе со всеми содержащимися в нем файлами. Разумеется, подтом предварительно нужно отключить от дерева каталогов.

```
root# umount /media/sub2
root# btrfs subvolume delete /media/btrfs/data/sub2
```

Но учтите и тот факт, что дисковое пространство, которое занимал подтом, освобождается не сразу после выполнения btrfs subvolume delete name, а постепенно. Высвобождением места при этом занимается специальный процесс ядра.

Мгновенные снимки

Мгновенные снимки (snapshots) позволяют разделить файловую систему btrfs или определенный ее подтом на две ветви. Новые мгновенные снимки можно создавать из самой файловой системы, из подтома или из другого мгновенного снимка.

Сразу после создания мгновенный снимок содержит ровно те же данные, что и исходный том. Впоследствии обе получившиеся файловые системы можно изменять

независимо друг от друга, При этом btrfs сохраняет только изменения. (Это означает, что при создании мгновенного снимка не копируется весь массив данных.)

Мгновенные снимки можно использовать, например, для резервного копирования. Сначала вы создаете мгновенный снимок, а потом сохраняете его как резервную копию. Таким образом, удается гарантировать, что при резервном копировании не изменяются никакие файлы. Одновременно с резервным копированием вы можете продолжать работу в обычном режиме. Как только резервное копирование завершается, соответствующий мгновенный снимок удаляется.

Еще один вариант применения мгновенных снимков связан с выполнением критически важных операций (например, обновления ядра) и заключается в создании «промежуточных сохранений». Если обновление не удастся довести до конца, то вы сможете продолжить его с момента, в который был выполнен мгновенный снимок.

Внутри системы btrfs мгновенные снимки обрабатываются как подтома. Поэтому большинство subvolume-команд системы btrfs применимы не только к подтомам, но и к мгновенным снимкам. Но существенное отличие между подтомами и мгновенными снимками заключается в том, что подтом изначально пуст, а мгновенный снимок, напротив, содержит виртуальную копию исходного каталога (Как уже говорилось выше, сами данные не копируются, а копируются лишь изменения по сравнению с исходным состоянием.)

Сравнение btrfs и LVM. Термин «мгновенный снимок» употребляется в контексте btrfs и применительно к диспетчеру логических томов (LVM) в двух совершенно разных значениях. В LVM мгновенный снимок — это неизменяемая копия логического тома. При создании такого мгновенного снимка необходимо указывать, сколько дискового пространства максимально может занимать такой мгновенный снимок. Эта информация требуется для того, чтобы при необходимости архивировать блоки данных исходного логического тома, прежде чем они будут изменены. Если дисковое пространство, отведенное под логический том, оказывается израсходованным, мгновенный снимок становится непригодным для использования.

Напротив, содержимое мгновенного снимка btrfs можно изменять! Такой мгновенный снимок представляет собой новое ответвление дерева каталогов, которое, на момент создания, совершенно идентично с этим мгновенным снимком. После этого обе ветви могут развиваться независимо друг от друга. Они требуют тем больше дискового пространства, чем больше файлов подвергается изменениям. Для сохранения изменений btrfs использует обычный btrfs-пул памяти. Но такая операция длится до тех пор, пока не будет обработана вся емкость файловой системы. Таким образом, мгновенные снимки отличаются гораздо большим функциональным разнообразием и гибкостью, нежели мгновенные снимки LVM!

Границы и проектные возможности. На первый взгляд кажется, что мгновенные снимки предлагают неограниченные возможности для администрирования файловой системы. Но на самом деле в их реализации в настоящий момент есть слабые места.

- Невозможно установить, сколько именно дискового пространства требует мгновенный снимок.

- Конечно, существует возможность задать мгновенный снимок в качестве нового тома по умолчанию (`btrfs subvolume set-default`), но нельзя удалить исходную базовую систему. Это ограничение касается, в первую очередь, применяемого в Fedora Yum-плагина (`yum-plugin-fs-snapshot`).

  Чтобы обойти это ограничение, нужно сразу после создания файловой системы `btrfs` сделать мгновенный снимок или подтом и сохранить там исходную систему. Таким образом, позже вы сможете удалить эту исходную систему в пользу мгновенного снимка.

  До сих пор все явные и неявные детали кажутся довольно запутанными, и в почтовой рассылке `btrfs` по этим вопросам идут активные дискуссии, например:

  - http://www.mail-archive.com/linux-btrfs@vger.kernel.org/msg03484.html;
  - http://www.mail-archive.com/linux-btrfs@vger.kernel.org/msg05334.html;
  - http://www.mail-archive.com/linux-btrfs@vger.kernel.org/msg04968.html.

Пример. Данный пример смоделирован на основе файловой системы `btrfs`, находящейся в сегменте `/dev/sdb1`. Файловая система подключается к дереву каталогов в точке `/media/btrfs`. Команда `btrfs subvolume snap` в данном случае создает мгновенный снимок всей файловой системы. Одновременно с этим создается каталог `/media/btrfs/snap1`. Через этот каталог можно использовать мгновенный снимок либо подключать данный мгновенный снимок с помощью команды `mount` к дереву каталогов в качестве отдельной файловой системы. При этом применяется `mount`–параметр `subvol=name`, уже рассмотренный выше.

```
root# btrfs subvolume snapshot /media/btrfs/ /media/btrfs/snap1
root# mkdir /media/snap1
root# mount -o subvol=snap1 /dev/sdb1 /media/snap1/
```

Теперь можно независимо работать как в исходной файловой системе, так и в ее мгновенном снимке (то есть без взаимного влияния двух вариантов друг на друга). При этом можно создавать, изменять и удалять файлы.

Как правило, мгновенные снимки `btrfs` поддаются изменениям. Если вы хотите создать мгновенный снимок только для чтения и использовать его в качестве резервной копии, выполните команду `btrfs subvolume snapshot` с параметром `-r`.

Распределение файловых систем btrfs на несколько устройств, RAID

Драйвер `btrfs` способен распределять файловые системы на несколько жестких дисков (или других устройств), при этом поддерживает RAID-уровни 0, 1 и 10, не прибегая к обычному драйверу Linux `mdadm`, предназначенному для работы с RAID (избыточным массивом независимых дисков). В будущем планируется реализовать в системе `btrfs` также поддержку RAID-5 и RAID-6.

Добавление устройства

Простейший случай, когда возникает необходимость распределить файловую систему на несколько устройств, обычно возникает тогда, когда имеющаяся файловая

система оказывается слишком мала. Ее вполне можно расширить за счет добавления нового устройства (то есть добавить пустой раздел жесткого диска или неиспользуемый логический том). Благодаря этому размер файловой системы увеличится. Вам не придется ни форматировать файловую систему, ни явно изменять ее размер — btrfs решит эти задачи самостоятельно.

```
root# btrfs device add /dev/sdb2 /media/btrfs
```

Сначала все файлы находятся на первом устройстве, а второе устройство заполняется постепенно. Если устройства располагаются на различных физических дисках (и только в этом случае), такой подход позволяет увеличить скорость работы — если распределить файлы с помощью команды btrfs filesystem balance по всем устройствам. Но при этом необходимо учитывать, что команда btrfs filesystem balance обычно выполняется очень долго и работа не стоит затраченных ресурсов.

```
root# btrfs filesystem balance /media/btrfs
```

Кроме того, устройство можно снова удалить. Данные, содержащиеся на устройстве, при этом сначала переносятся на другие устройства, и поэтому выполнение следующей команды может длиться очень долго:

```
root# btrfs device delete /dev/sdb1 /media/btrfs/
```

ПРИМЕЧАНИЕ

В ходе тестов, которые я проводил летом 2012 года (с версией ядра 3.5), команда btrfs device delete аварийно завершалась с ошибкой invalid argument (недействительный аргумент). Однако раньше эта команда работала. Поэтому можно надеяться, что вскоре данная ошибка будет исправлена.

mkfs.btrfs с несколькими устройствами

Можно сразу создать файловую систему btrfs на нескольких устройствах. При этом вы передаете команде mkfs.btrfs имена нескольких устройств:

```
root# mkfs.btrfs /dev/sdb1 /dev/sdc1
```

По умолчанию при этом метаданные файловой системы дублируются (это соответствует RAID-1), но сами данные распределяются на несколько устройств. Метаданные содержат информацию об управлении файловой системой, например списки индексных дескрипторов, а также деревья для поиска по файлам. К сожалению, в результирующей файловой системе скорость работы оставляет желать лучшего (работа протекает медленнее, чем в RAID-0, это объясняется дублированием метаданных). Кроме того, что касается безопасности, такая система не может сравниться с RAID-1, поскольку в описываемом случае не происходит избыточного сохранения информации.

При выполнении команды mount задается любое желаемое устройство файловой системы. (После перезапуска компьютера необходимо выполнить команду btrfs device scan, чтобы сообщить файловой системе btrfs, какие устройства с файловой системой btrfs есть в наличии и как они взаимосвязаны.)

```
root# mount -t /dev/sdb1 /media/btrfs
```

Чтобы узнать, какой вариант RAID применяется с каким видом данных, выполните команду btrfs filesystem:

```
root# btrfs filesystem df /media/btrfs
Data, RAID0: total=1.56GB, used=128.00KB
System, RAID1: total=8.00MB, used=4.00KB
Metadata, RAID1: total=1.00GB, used=24.00KB
...
```

RAID-0

Если вы хотите создать так называемую настоящую систему RAID, в которой и данные, и метаданные обрабатываются по одному принципу, то сообщите желаемый уровень RAID команде mkfs.btrfs с параметром -d (для данных) или -m (для метаданных). Кроме того, укажите mkfs.btrfs желаемое количество устройств. Следующая команда создает систему RAID-0 (с распределением данных):

```
root# mkfs.btrfs -d raid0 -m raid0 /dev/sdb1 /dev/sdc1
```

RAID-1

Аналогичным образом создается и файловая система избыточного массива независимых дисков RAID-1:

```
root# mkfs.btrfs -d raid1 -m raid1 /dev/sdb1 /dev/sdc1
```

Интересный случай возникает при отказе одного из устройств. Чтобы протестировать такой случай, я удалил жесткий диск /dev/sdc. Чтобы файловую систему снова можно было использовать, следует применить дополнительный mount-параметр degraded:

```
root# mount -o degraded /dev/sdb1 /media/btrfs
```

Чтобы восстановить массив RAID, добавьте в файловую систему новое устройство — желательно, такого же размера, как и удаленное устройство. В следующем примере я снова добавил /dev/sdc1, но этот раздел взят уже с нового жесткого диска. Чтобы снова распределить файловую систему на два устройства и восстановить, таким образом, избыточность RAID-1, необходимо выполнить команду btrfs filesystem balance. Естественно, в крупных файловых системах выполнение этой команды длится довольно долго. Правда, при этом файловую систему все равно можно использовать (пусть даже и со значительно сниженной скоростью работы).

```
root# btrfs device add /dev/sdc1 /media/btrfs
root# btrfs filesystem balance /media/btrfs
```

И только после этого неисправное устройство можно удалить из файловой системы. В таком случае устройство указывается с применением ключевого слова missing:

```
root# btrfs device delete missing /media/btrfs
```

При следующем выполнении команды mount можно не задавать параметр degraded.

Определение того, что используется файловая система btrfs (df)

При работе с другими файловыми системами можно просто ввести `df -h` и узнать, сколько дискового пространства есть в вашем распоряжении, какая доля из него занята и какая — свободна. Но при использовании файловых систем `btrfs` команда `df` выдает совершенно неверные результаты, особенно если вы применяете RAID. Такие ошибки возникают по трем причинам.

- В настоящее время `btrfs` не сообщает `df` никакой информации о том, как именно используются устройства (то есть, например, не передает уровень RAID). Получить такую информацию невозможно не в последнюю очередь потому, что `btrfs` отводит для работы с данными и метаданными разные уровни RAID.
- `btrfs` разделяет использование памяти на три различные части: системные данные (это совсем небольшие множества данных), метаданные (контрольные суммы и информация об управлении, которые позволяют ускорить поиск по файлам) и полезные данные (пространство, занимаемое самим содержимым файлов).
- `btrfs` сразу требует для себя не только все доступное дисковое пространство в разделе, но и использует долговременную память в качестве пула, из которого резервирует по мере необходимости блоки данных для системной информации, метаданных или полезных данных. При этом бросается в глаза тот факт, что `btrfs` задействует относительно много метаданных. Метаданные применяются преимущественно для хранения контрольных сумм файлов.

К сожалению, такие команды `btrfs`, как `filesystem show` и `filesystem df`, также не в состоянии точно определить, сколько именно места на диске еще свободно. Следующий пример должен помочь вам как минимум правильно интерпретировать данные, выдаваемые `btrfs`. В качестве исходной точки берем небольшую файловую систему `btrfs`-RAID-1, у нас она состоит из двух разделов, каждый из которых имеет размер 8 Гбайт. В этой файловой системе будет находиться весь скопированный нами каталог `/usr` (на него потребуется примерно 2,1 Гбайт дискового пространства).

```
root# mkfs.btrfs -d raid1 -m raid1 /dev/sdb1 /dev/sdc1
root# mount /dev/sdb1 /media/btrfs
root# cp -r /usr /media/btrfs
```

df

`df` дает совершенно неверный результат. Это объясняется тем, что `df` не в состоянии опознать, что `btrfs` работает на уровне RAID-1. Общий объем рассчитывается как сумма объемов разделов. Тот объем, который `df` считает используемым, на самом деле просто зарезервирован для системы `btrfs`.

```
root# df -h /media/btrfs/
Файловая система Размер Использовано Доступно. Использовано % Подключено к
/dev/sdb1        16G    3,7G         12G       25%            /media/btrfs
```

filesystem show

Команда `btrfs filesystem` сообщает, что файловая система распределена на два устройства, каждое из которых имеет размер 8 Гбайт. Общий объем системных данных, полезной информации и метаданных, содержащийся в системе, — 1,9 Гбайт. Кроме того, мы теперь знаем, что на каждом устройстве зарезервировано примерно по 3,5 Гбайт данных. Команда `filesystem show` ничего не сообщает о том, какова фактическая связь между устройствами (то есть, какой уровень RAID активен в настоящий момент). Поэтому мы не можем судить, каков общий объем файловой системы (при RAID-0 он составил бы 16 Гбайт, а при RAID-1 — всего 8 Гбайт).

```
root# btrfs filesystem show /dev/sdb1
Label: none uuid: dc691a5d-187e-4cb4-a94a-d12dabdffde4
      Total devices 2 FS bytes used 1,89GB
      devid 1  size   7.81GB used 3,53GB path /dev/sdb1
      devid 2  size   7.81GB used 3,54GB path /dev/sdc1
```

filesystem df

Команда `filesystem df` сообщает о том, как именно применяются зарезервированные данные. `btrfs` уже зарезервировала под полезные данные 3 Гбайт, но в действительности использовала всего около 1,6 Гбайт из них.

Далее видим, что `btrfs` зарезервировала под метаданные 512 Мбайт и использовала из них ровно 190 Мбайт. Наконец, еще 32 Мбайт `btrfs` зарезервировала под системные данные, и из этого объема уже задействовала всего 4 Кбайт. 3,00 + 0,5 + + 0,03 = 3,53 Гбайт, именно эту величину и отобразила команда `filesystem show`.

```
root# btrfs filesystem df /media/btrfs/
Data, RAID1:       total=3.00GB, used=1.62GB
System, RAID1:     total=32.00MB, used=4.00KB
System:            total=4.00MB, used=0.00
Metadata, RAID1:   total=512.00MB, used=191.02MB

Data:     total=2.61GB, used=1.70GB
Metadata: total=1.01GB, used=198.82MB
System:   total=12.00MB, used=4.00KB
```

Сколько же свободного места действительно есть у нас в распоряжении? Мы знаем, что общий объем дискового пространства — 16 Гбайт. Из-за избыточности, связанной с использованием RAID-1, полезное пространство уменьшается наполовину, то есть составляет 8 Гбайт. Из этого объема уже зарезервировано 3,53 Гбайт. Итак, область, отведенная под полезные данные, может быть увеличена еще максимум на 4,3 Гбайт и составит ровно 7 Гбайт. И уже 1,6 Гбайт из этих 7 использовано. В итоге, имеем, что свободное пространство, которые мы еще можем заполнить нашими файлами, составляет около 5,4 Гбайт. В любом случае эти расчеты верны лишь при условии, что `btrfs` в ходе работы не попытается зарезервировать дополнительное место для метаданных.

Как видите, вычислить объем свободного дискового пространства не так просто. Если в будущем `btrfs` будет поддерживать еще и RAID-5 и RAID-6, ситуация еще более усложниться. При этом в рассылке `btrfs` снова и снова появляются сообщения

о том, что btrfs показывает, что свободное пространство израсходовано, хотя на диске должно быть свободно еще немало места. Подытожим: работая с файловой системой btrfs, рекомендуется оставлять большой резерв места на диске.

14.9. Файловая система xfs

Файловая система xfs была разработана в 1994 году фирмой SGI для ее рабочей станции, действовавшей на базе UNIX-подобной файловой системы IRIX. Позже эта файловая система была адаптирована для Linux, а теперь официально входит в состав ядра (версии 2.6 и выше). Система считается хорошо продуманной, стабильной и удобной, в первую очередь, для работы с очень большими (мультимедийными) файлами. Более подробная информация о файловой системе xfs сообщается здесь:

- man xfs;
- http://en.wikipedia.org/wiki/XFS;
- http://xfs.org/index.php/XFS_FAQ.

ВНИМАНИЕ

При работе с xfs не забывайте о двух важных особенностях. Во-первых, файловые системы xfs можно увеличивать командой xfs_growfs, а уменьшать нельзя. Во-вторых, эта система использует раздел, начиная с первого байта и, в отличие от других файловых систем Linux, не оставляет места для загрузочного сектора. Поэтому при установке GRUB в загрузочный сектор раздела системы xfs уничтожаются части этой файловой системы! Если вам требуется установить GRUB, делайте это в ведущем секторе или в загрузочном секторе другого раздела.

/etc/fstab. Записи для рассматриваемой файловой системы в каталоге /etc/fstab обычно выглядят, как в следующем примере. Дополнительные параметры mount используются очень редко. Они перечислены в man mount.

```
# /etc/fstab
/dev/sdb13   /data   xfs   defaults  0  0
```

Создание файловой системы xfs. Чтобы создать в разделе диска файловую систему xfs, просто выполните команду mkfs.xfs:

```
root# mkfs.xfs /dev/sdc1
meta-data=/dev/sdc1    isize=256    agcount=16, agsize=152742 blks
         =             sectsz=512   attr=0
data     =             bsize=4096   blocks=2443872, imaxpct=25
         =             sunit=0      swidth=0 blks, unwritten=1
naming   =version 2    bsize=4096
log      =internal log bsize=4096   blocks=2560, version=1
         = sectsz=512  sunit=0 blks
realtime =none         extsz=65536  blocks=0, rtextents=0
```

Осталось выполнить команду mount и можно приступать к работе с файловой системой.

```
root# mount -t xfs /dev/sdc1 /test
```

Проверка файловой системы. Целостность файловой системы xfs автоматически проверяется при каждой операции `mount` (для этого просто интерпретируется протокол журналирования). Чтобы проверить систему вручную, выполните команду `xfs_check`. Это можно сделать, когда файловая система еще не подключена. Если команда обнаружит ошибки, можете попытаться исправить их с помощью команды `xfs_repair`.

Если хотите обеспечить совместимость с другими файловыми системами, применяйте команду `fsck.xfs`. Однако помните, что она не выполняет никаких задач и всегда возвращает результат `OK`.

Изменение параметров файловой системы. Команда `xfs_growfs` позволяет увеличить файловую систему xfs на ходу (файловая система должна быть подключена к дереву каталогов!). Для выполнения команды необходимо, чтобы раздел диска, в котором расположена файловая система, уже был увеличен. Команда `xfs_admin` позволяет изменить различные параметры файловой системы, например ее название (`Label`) и уникальный идентификационный номер. Для этого файловую систему сначала нужно отключить от дерева каталогов (`unmount`).

14.10. Файловые системы Windows (VFAT, NTFS)

У многих пользователей Linux на компьютере параллельно установлена версия Windows. Внешние носители данных (USB-флешки, карты памяти от цифровых фотоаппаратов) также часто используют файловые системы Windows. Далее будет рассказано, как, работая в Linux, обращаться к данным, сохраненным в файловых системах Windows, — независимо от того, находится эта информация в одном из разделов внутреннего жесткого диска либо на внешнем носителе.

Варианты. Строго говоря, есть два типа файловых систем Windows.

- **FAT, VFAT, exFAT** — существуют многочисленные варианты FAT. Раньше других возникли FAT12 для дискет, FAT16 для файловых систем размером менее 2 Гбайт, а также FAT32 для файловых систем размером до 8 Тбайт и для файлов до 4 Гбайт.

  В Windows 95 появилось дополнение VFAT, которое позволило сохранять во всех вариантах FAT названия, превышающие лимит 8 + 3 символов, известный со времен MS-DOS. Длинные названия файлов система сохраняет как последовательности символов в Unicode. В более новых версиях Windows все имена файлов (в том числе короткие) сохраняются в таком виде. Подобный метод позволяет гарантировать, что регистр названий файлов сохранится и не возникнет никаких проблем с кодировками из-за применения различных кодовых страниц.

  Обычный раздел Windows, располагающийся на жестком диске, чаще всего содержит комбинацию FAT32 и VFAT (VFAT32). В дальнейшем, говоря о VFAT (или используя запись `mount/fstab` при обозначении файловой системы), я буду иметь в виду любые комбинации из FAT12/16/32 и VFAT.

Существует относительно новый вариант FAT — система exFAT, которая была разработана для масштабных операций с Flash-памятью. В этой системе могут существовать файлы размером до 16 777 216 Тбайт, поддерживаются списки контроля доступа и транзакции. В настоящее время в Linux предоставляется бета-версия соответствующего свободно распространяемого драйвера (http://code.google.com/p/exfat/), а также коммерческий драйвер фирмы Tuxera (http://tuxera.com/). Но из-за проблем, связанных с патентами, интеграция драйвера exFAT в ядро, к сожалению, невозможна.

- **NTFS (New Technology File System)** — появилась в Windows NT и поддерживается во всех современных версиях Windows. По сравнению с FAT, NTFS отличается повышенной надежностью (права доступа, журналирование и т. д.), а также содержит разнообразные дополнительные функции. Размер файловой системы практически не ограничен (16 777 216 Тбайт).

Поддержка в Linux. Linux способна работать с системами (V)FAT и NTFS — считывать из них информацию и вносить изменения. В файловой системе exFAT есть временный драйвер для Linux «только для чтения», пока не входящий в состав ядра.

ВНИМАНИЕ

Если вы хотите внести изменения в файловую систему Windows на компьютере, поддерживающем мультисистемную загрузку, то сначала полностью завершите работу Windows (а не переводите систему в спящий или ждущий режим) и только потом запустите Linux. Иначе вы можете спровоцировать несовместимость файловой системы и потерять данные.

В Linux почти никогда не приходится создавать файловые системы Windows, хотя такое возможно. С помощью команды `mkfs.vfat` можно отформатировать раздел диска для системы VFAT, используя `mkfs.ntfs` — для NTFS (обычно `mkfs.ntfs` находится в пакете `ntfsprogs`, который требуется дополнительно установить).

Преобразование текстовых файлов. Независимо от применяемой файловой системы обмен текстовыми файлами между Linux и Windows вызывает проблемы, так как в каждой из операционных систем применяются различные кодировки и обозначения конца строки. Эти проблемы решаются с помощью инструментов конвертации.

Права доступа. В VFAT вообще не существует концепции прав доступа. В NTFS права доступа хотя и поддерживаются, но иначе, чем в UNIX/Linux. Возникает следующий вопрос: какие пользователи Linux получат определенные права доступа к файлам Windows? Ответ на этот вопрос дают параметры `uid`, `gid` и `umask`/`fmask`/`dmask`. Они определяют владельцев, их групповую отнесенность и биты доступа к файловой системе Windows — права для всех файлов этой системы будут одинаковы, независимо от того, как определены права доступа NTFS.

Файловая система VFAT

Стандартные настройки. Для начала коротко обобщим стандартные настройки драйвера файловой системы VFAT: драйвер самостоятельно распознает тип FAT (FAT12/-16/-32). Имена файлов Windows в Linux отображаются в кодировке

латиница-1 (ISO8859-1). Пользователь, выполняющий команду `mount`, имеет право читать все файлы во всех каталогах, а также вносить изменения в эти файлы. Все остальные пользователи имеют право читать файлы, но не могут вносить изменения.

/etc/fstab. Как правило, запись в `/etc/fstab` для раздела локального диска с системой VFAT выглядит так:

```
# /etc/fstab
/dev/sda1    /media/win1    vfat    utf8,uid=1000   0 0
```

В результате имеем: пользователь с номером 1000 имеет право изменять все файлы, а специальные символы, содержащиеся в длинных именах файлов Windows (включающие более 8 + 3 символов), отображаются в Linux в кодировке UTF-8.

Следующая строка `fstab` не подключает раздел с Windows к дереву каталогов автоматически (`noauto`). Однако благодаря `users` каждый пользователь может выполнить команду `mount`. Кроме того, файлы Windows относятся к группе, являющейся стандартной для определенного пользователя, а не к группе, актуальной в настоящий момент.

```
# /etc/fstab
/dev/sda1    /media/win1    vfat    noauto,users,gid=users,utf8   0  0
```

Файловая система NTFS (ntfs-3g)

Раньше существовало много разных NTFS-драйверов для Linux. В последнее время определился доминирующий драйвер — ntfs-3g. Он поддерживает доступ для чтения и изменения файлов, а также может работать с потоками. Правда, этот драйвер не позволяет читать и изменять зашифрованные файлы, а также архивировать файлы (хотя дает возможность читать архивированные файлы). Почти во всех крупных дистрибутивах драйвер ntfs-3g устанавливается по умолчанию.

В отличие от большинства других драйверов файловых систем, ntfs-3g внедряется не как модуль ядра, а как FUSE-драйвер. FUSE означает Filesystem in Userspace (Файловая система в пользовательском пространстве). Это модуль ядра, обменивающийся информацией с внешними программами. Иными словами, FUSE позволяет установить драйвер файловой системы вне ядра.

Файл /etc/fstab. Как правило, строка `fstab`, служащая для автоматического интегрирования раздела NTFS в дерево каталогов, выглядит так:

```
# /etc/fstab
/dev/sda1   /media/win   ntfs-3g   uid=1000,gid=1000  0 0
```

В большинстве дистрибутивов можно обозначить файловую систему просто NTFS (а не ntfs-3d).

Потоки. Потоки — характерная черта файловой системы NTFS: файл NTFS может состоять из нескольких потоков. При этом каждый поток функционально аналогичен обычному файлу. При простом доступе к файлу автоматически считывается или изменяется стандартный поток.

При работе с драйвером ntfs-3g параметр `streams_interface` управляет доступом к потокам. При стандартных настройках `xattr` потоки обрабатываются как атрибуты

файла. Доступ к потокам обеспечивают команды get или setfattr из пакета attr. Команда getfattr -d -e text возвращает список всех атрибутов, причем их содержимое представляется в виде текста.

```
root# mount /dev/sda1 /media/win
root# cd /media/win
root# cat > streamtest
   abc (Ctrl)+(D)
root# setfattr -n user.stream1 -v "efg" streamtest
root# setfattr -n user.stream2 -v "xyz" streamtest
root# cat streamtest
abc
root# getfattr -d -e text streamtest
# file: streamtest
user.stream1="efg"
user.stream2="xyz"
root# cd
root# umount /media/win
```

Кроме того, можно работать с настройкой streams_interface=windows. Она активизирует типичный для Windows метод записи в форме *имя_файла*:*имя_потока*.

```
root# mount -o streams_interface=windows /dev/sda1 /media/win
root# cd /media/win
root# cat streamtest
abc
root# cat streamtest:stream1
efg
```

Администрирование. В пакете ntfsprogs содержатся различные команды, помогающие при администрировании файловых систем NTFS (табл. 14.8).

Таблица 14.8. Команды из пакета ntfsprogs

Команда	Значение
mkntfs	Создает файловую систему NTFS
ntfsclone	Копирует файловую систему NTFS
ntfsinfo	Сообщает информацию о файловой системе NTFS
ntfslabel	Дает заголовок разделу с NTFS
ntfsresize	Изменяет размер файловой системы NTFS
ntfsundelete	Пытается восстановить удаленные файлы

14.11. CD, DVD, дискеты

CD и DVD с данными

В принципе управление CD и DVD не отличается от управления жесткими дисками. Однако здесь есть два важных отличия: во-первых, диск в приводе можно заменить, а обычный жесткий диск заменить на ходу нельзя. Во-вторых,

на CD и DVD с данными используются особые файловые системы: ISO9660 или UDF.

ISO9660 и UDF. Сначала коротко рассмотрим эти файловые системы в целом. *ISO9660* представляет собой общепризнанный стандарт для CD с данными. Поскольку эта система имеет несколько фундаментальных ограничений, для нее известны также некоторые расширения. Характерное для UNIX расширение Rockridge позволяет сохранять длинные названия файлов и права доступа. Характерное для Windows расширение Joliet обеспечивает возможность использовать в именах файлов символы Unicode. Расширение El-Torito позволяет запускать компьютер прямо с CD.

UDF (универсальный дисковый формат) появился после ISO9660. Он используется на многих DVD (хотя вполне может применяться и формат ISO9660). Отличия от ISO9660 заключаются в том, что файлы могут иметь размер более 2 Гбайт, имена файлов без установки каких-либо дополнений могут состоять из символов Unicode общим количеством до 255, лучше поддерживаются RW-носители (пакетная запись) и т. д.

Названия устройств CD/DVD. В табл. 14.9 указано, какие названия устройств применяются при доступе к приводу CD/DVD. Название устройства зависит от того, как подключен привод (IDE, SCSI, SATA, USB или Firewire) и применяется ли современное ядро с поддержкой libata. Вероятнее всего, вам встретится `/dev/scd0`. Только в очень старых компьютерах вам, может быть, попадется `/dev/hda`, `/dev/hdb` и т. д. Кроме того, существуют файлы-устройства `/dev/cdrom`, `/dev/dvd` или `/dev/dvd-recorder`. В данном случае мы имеем дело со ссылками на настоящие файлы-устройства.

Таблица 14.9. Названия устройств CD/DVD

Название устройства	Значение
/dev/scd0 или /dev/sr0	Первый привод CD/DVD
/dev/scd1 или /dev/sr1	Второй привод CD/DVD

Автоматическая эксплуатация. В большинстве дистрибутивов конфигурация локальной системы задается таким образом, что при вкладывании в привод CD или DVD автоматически открывается окно файлового менеджера, отображающее содержимое носителя. В любой момент можно извлечь CD/DVD, нажав кнопку на приводе или щелкнув на значке соответствующего привода в контекстном меню. Такое удобство обеспечивается благодаря «закулисному» управлению оборудованием в Linux и работе специальной службы KDE или демона Gnome.

Ручное управление. Если вы работаете с консолью либо с локальной системой, где не предусмотрено автоматическое управление CD/DVD, вам потребуется вручную подключать диск к дереву каталогов после того, как он окажется в приводе. Как обычно, названия устройств и каталогов при этом могут варьироваться, в зависимости от оборудования и дистрибутива.

```
root# mount -t iso9660 -o ro /dev/scd0 /media/dvd   (ISO9660-CD/DVD)
root# mount -t udf -o ro /dev/scd0 /media/dvd       (UDF-DVD)
```

По умолчанию все файлы и каталоги открыты для чтения всем пользователям. Если вы хотите непосредственно запускать программы, находящиеся на CD или

DVD, добавьте параметр exec. Чтобы интернациональные имена файлов правильно обрабатывались, нужно использовать параметр iocharset=utf8 или просто utf8.

Прежде чем извлечь CD/DVD, необходимо выполнить команду unmount:

```
root# umount /media/dvd
```

СОВЕТ

Вместо unmount вы можете выполнить eject. В таком случае CD не только отключается от файловой системы, но и извлекается из компьютера. Если в компьютере находится несколько носителей данных, которые можно извлечь (CD, DVD, ЗУ на магнитной ленте), эти варианты интерпретируются по очереди: извлекается первый в очереди носитель данных. Вы также можете задать нужное устройство, указав название устройства или точку подключения к системе.

Device is busy (Устройство занято). Если команда unmount возвращает ошибку Device is busy (Устройство занято), это означает, что данные с CD-ROM используются другой программой. Такая реакция возникает и в том случае, если какой-либо из каталогов диска открыт в одной из оболочек. Выполните в нем команду cd, чтобы перейти в домашний каталог. Если требуется найти процесс, из-за которого возникает такая ошибка, воспользуйтесь командой fuser — выполните fuser -m /cdrom.

Еще одной причиной такой ошибки может быть NFS: если привод CD-ROM используется через NFS на другом компьютере, выполнить unmount зачастую не удается и тогда, когда тот компьютер уже давно открыл доступ к приводу. В таких случаях требуется перезагрузить NFS-сервер и (иногда) даже сам компьютер.

/etc/fstab. В большинстве дистрибутивов применяется описанное выше автоматическое управление, и поэтому в /etc/fstab отсутствует запись, касающаяся CD/DVD-привода (она не нужна). Однако если вы часто вручную подключаете CD/DVD к дереву каталогов, такая запись вам пригодится. Она будет выглядеть примерно как в следующем образце:

```
# /etc/fstab
/dev/scd0   /media/dvd  udf,iso9660   users,noauto,ro   0  0
```

Теперь достаточно будет команд mount /media/dvd или mount /media/dvd, чтобы интегрировать CD/DVD в дерево каталогов либо отключить диск от дерева. Эти команды может выполнять любой пользователь.

AudioCD, VideoDVD

AudioCD. Работа с AudioCD отличается от обработки CD с данными. Они не подключаются к дереву каталогов командой mount, а напрямую считываются специальными программами (например, в KDE и Gnome это программы Amarok и Rhytmbox соответственно). Можно выполнить и цифровое считывание аудиотреков (например, для последующего преобразования их в файлы Ogg-Vorbis).

VideoDVD. Как правило, при работе с VideoDVD используется файловая система UDF.

Запись CD и DVD. Для записи CD и DVD применяйте в KDE программу K3B, в Gnome — Brasero, а в консоли — команду wodim.

Дискеты

При повседневной работе с Linux дискеты уже не играют никакой роли. Однако если вы вдруг окажетесь в ситуации, когда нужно считать информацию с дискеты, вам потребуется подключить ее к файловой системе с помощью `mount` и как обычно обратиться к ее файлам. Для дискет чаще всего используется название устройства `/dev/fd0`. На них могут находиться различные файловые системы. Обычно дискеты для MS-DOS/Windows работают с файловой системой VFAT.

```
root# mount -t vfat /dev/fd0 /media/floppy
```

С помощью команд из пакета `mtools` (например, `mcopy` или `mdir`) можно считывать информацию с дискет MS-DOS/Windows либо изменять ее и без применения `mount`. Командой `fdformat` производится низкоуровневое форматирование. Чтобы дополнительно создать файловую систему, выполните команду `mkfs.vfat`.

14.12. Внешние носители данных (USB, Firewire и др.)

USB-флешки, карты памяти цифровых фотоаппаратов, жесткие диски Firewire и eSATA, а также другие внешние носители имеют важный общий признак: они на ходу подключаются к компьютеру, на ходу же и отключаются. Система работает почти со всеми такими носителями как с SCSI.

Автоматическое управление. Локальные системы (KDE, Gnome) практически всех дистрибутивов реагируют на подключение внешнего носителя так: открывается новое окно файлового менеджера (иногда с запросом о подтверждении), обеспечивающее удобный доступ к файлам подключенного внешнего носителя. Часто на Рабочем столе появляется значок, обозначающий носитель и позволяющий открыть контекстное меню, через которое файловую систему можно специально отключить от дерева каталогов.

ВНИМАНИЕ

Необходимо специально отключать все разделы внешнего носителя из дерева каталогов, а только потом извлекать кабель! В большинстве дистрибутивов для этого нужно щелкнуть на значке носителя и выбрать Eject (Извлечь), Safely Remove (Безопасное извлечение) или подобную запись в меню. Таким образом вы гарантируете, что все операции записи будут завершены, а потом на самом деле отключится устройство. Если пренебречь этим шагом, вы рискуете повредить файловую систему и потерять данные!

В KDE и Gnome возможна ситуация, в которой несколько пользователей параллельно входят в систему. В таком случае права доступа к новому подключенному внешнему носителю, как правило, получает пользователь, вошедший в систему раньше других. Этот частный случай по-разному решается в различных дистрибутивах (или вообще не решается), поэтому могут возникнуть проблемы.

Таким образом, старайтесь не менять пользователя, когда работаете с внешними носителями!

Внутрисистемная обработка горячего подключения. Управление горячим подключением в современных дистрибутивах осуществляется в тесном взаимодействии ядра, системы udev, системы обмена информацией D-Bus и программы PolicyKit. В более старых версиях вы, возможно, встретите программы `supermount`, `magicdev` или `subfs/submount`, но все они не очень хорошо работают.

Управление вручную. При работе в текстовом режиме или с локальной системой, в которой не предусмотрено автоматическое управление носителями, вам потребуется самостоятельно выполнить команду `mount`. Для этого сначала определите, какое название имеет ваше устройство (как правило, это `/dev/sd`*x*, где *x* — первая свободная буква по алфавиту).

Обзор всех носителей данных (включая жесткие диски, но исключая приводы CD и DVD) выводит команда `fdisk -l`. В следующем примере `/dev/sdf1` — первый и единственный раздел на USB-флешке.

```
root# fdisk -l
...
Диск /dev/sdf: 256 MB, 256901120 байт
16 головок, 32 секторов/дорожек, 980 цилиндров
Единицы = цилиндры по 512 * 512 = 262144 байт
Device     Boot    Start    End    Blocks    Id   System
/dev/sdf1     *        1    980    250864    e    W95 FAT16 (LBA)
```

USB-флешки и карты памяти также можно форматировать как Superfloppy. Это означает, что на диске не создается таблица разделов. В таком случае весь привод запрашивается как устройство `/dev/sda` (вместо обычного способа именования устройства с помощью `/dev/sda1`, когда указывается номер определенного раздела).

Если знать номера устройств, дальше все просто: создается новый каталог и выполняется следующая команда `mount`:

```
root# mkdir /media/memorystick
root# mount /dev/sdf1 /media/memorystick
```

На внешних носителях могут использоваться файловые системы различных типов. На практике на внешних жестких дисках и USB-флешках чаще всего применяется система VFAT. То же касается карт памяти для различных электронных приборов.

После того как вы считаете или запишете все нужные файлы, выполните команду `unmount` как обычно. Ни в коем случае не отключайте кабель USB или Firewire, пока не выполните `unmount`, иначе рискуете потерять данные!

```
root# umount /media/memorystick
```

Файл /etc/fstab. Лишь администратор вправе выполнять команду `mount`. Если обычные пользователи должны иметь возможность самостоятельно подключать к дереву каталогов внешние носители и отключать их из системы, то вам потребуется вставить в `fstab` соответствующую строку с параметром `users`. Для USB-флешки с файловой системой VFAT эта строка может выглядеть так:

```
# /etc/fstab: USB-Stick
/dev/sdf1  /media/memorystick  vfat  users,gid=users,utf8,noatime,noauto 0 0
```

Теперь каждый пользователь может подключить USB-флешку к дереву каталогов с помощью команды `mount /media/memorystick`, а потом читать и изменять данные, содержащиеся на ней. Однако для этого метода характерны два серьезнейших недостатка.

- В зависимости от того, в каком порядке подключаются устройства, их названия изменяются. Если USB-флешка подключается в качестве второго или третьего устройства, ее название может быть, например, `/dev/sdg` и доступ к устройству через каталог `/media/memorystick` будет закрыт.
- Наоборот, вышеуказанная запись `fstab` может быть использована для доступа к другому устройству, что, возможно, не планировалось.

Таким образом, лучше всего задавать имя устройства в `/etc/fstab` не непосредственно, а через ссылку `by-uuid`. Чтобы узнать номер UUID, выполните команду `blkid`:

```
root# blkid /dev/sdf1
/dev/sdf1: UUID="4550-9BD2" TYPE="vfat"
```

Соответствующую запись `fstab` мы разделим на две строки, так как она не помещается на одной:

```
# /etc/fstab: USB-Stick
/dev/disk/by-uuid/4550-9BD2 /media/memorystick vfat \
                  users,gid=users,utf8,noatime,noauto 0 0
```

В принципе вы можете начать строку `fstab` с записи `UUID=4550-9BD2`. Команда `mount` в таком случае будет работать, как в прошлом примере, но с `unmount` возникнут проблемы: вместо уникального идентификатора UUID команда `mount` запишет носитель данных в `/etc/mtab` с актуальным названием устройства. При этом у `unmount` строки `/etc/fstab` и `/etc/mtab` не совпадут, в результате чего возникнет ошибка.

14.13. Разделы и файлы подкачки

Если оперативной памяти недостаточно для выполнения всех программ и в вашем распоряжении есть файлы и разделы подкачки, Linux задействует их в качестве дополнительной памяти *(страничная организация памяти)*. Таким образом, Linux может использовать больше памяти, чем доступно в RAM.

Обычно раздел подкачки создается при установке. Чтобы проверить, есть ли в вашем распоряжении виртуальная память и, если есть, то сколько (имеется или действительно применяется), выполните команду `free`. В следующем примере имеем 1519 Мбайт RAM и 2000 Мбайт виртуальной памяти. В настоящее время, согласно примеру, используется 401 Мбайт RAM для работы с программами и данными, остаток применяется как файловый буфер (кэш). Виртуальная память пока не задействуется.

```
root# free -m
             total       used       free     shared    buffers     cached
Mem:          1519       1479         39          0         67       1010
```

```
-/+ buffers/cache:      401      1117
Swap:          2000        0      2000
```

Если компьютер работает длительное время, ему рано или поздно придется использовать виртуальную память, даже если у вас в распоряжении еще достаточно RAM. Причина проста: ядро управляет кэшем для предоставления доступа к файлам «для чтения»; если позже понадобится какой-либо файл, его можно считать из кэша. Как только кэш превышает по размеру свободную оперативную память, Linux выгружает блоки памяти, которые давно не применялись, в раздел подкачки. Это совсем не означает, что осталось мало памяти. Linux просто пытается использовать имеющуюся память как можно более эффективно.

Файл /etc/fstab. Далее показаны две записи о разделах подкачки; эти записи находятся в файле `/etc/fstab`. Благодаря параметру `pri` оба раздела Linux используются равномерно. Это повышает скорость системы (как при работе с распределением данных (striping) или RAID-0), если разделы находятся на двух взаимно независимых жестких дисках. Если имеется только один раздел подкачки, укажите `defaults` вместо `pri=0`.

```
# /etc/fstab: разделы подкачки
/dev/sda9  swap   swap   pri=1   0  0
/dev/sdc7  swap   swap   pri=1   0  0
```

Управление работой виртуальной памяти. Если памяти в RAM начинает не хватать, ядро Linux применяет достаточно сложный алгоритм, позволяющий решить, следует ли открыть доступ к кэш-памяти для решения новых задач либо лучше выгрузить в раздел подкачки последние неиспользованные области памяти. Параметр ядра `/proc/sys/vm/swappiness` позволяет вам указать, что должно делать ядро в случае выбора — уменьшить кэш или выгрузить данные (что такое параметры ядра и как их настраивать, рассказано в разделе 17.5).

Стандартная настройка для `swappiness` равна `60`, диапазон возможных значений: от `0` до `100`. Значение `0` означает, что ядро по возможности не будет использовать *постраничную организацию памяти*; `100` говорит о том, что области памяти, не применяемые долгое время, будут отправляться в раздел подкачки как можно раньше. Более подробно параметр `swappiness` описан на сайтах http://lwn.net/Articles/83588/ и http://kerneltrap.org/node/3000.

На практике вы заметите работу разделов подкачки только тогда, когда оставите компьютер работать на всю ночь и за это время программа обратится к большому количеству файлов на диске (это может быть, например, сценарий для резервного копирования или программа для составления поискового индекса). Поскольку обращений к файлам много, кэш значительно увеличивается. Если в системе действует настройка `swappiness=60` (или выше), то ядро часами будет выгружать неиспользуемую память. Например, это может касаться страниц виртуальной памяти OpenOffice или Gimp. Если на следующий день вы захотите продолжить работу с OpenOffice, потребуется пара секунд, чтобы все эти страницы вернулись из раздела подкачки обратно в оперативную память. Если установить значение `swappiness=0`, то ждать в таком случае не придется.

Сколько нужно виртуальной памяти. Раньше часто можно было встретить рекомендацию оставлять на виртуальную память примерно в два раза больше места,

чем есть в оперативной. Однако с ростом объемов памяти это практическое правило действует все реже. Если вы работаете с Linux в основном на локальном компьютере, вам вполне хватит и относительно небольшого раздела подкачки (например, раздел подкачки размером 512 Мбайт при оперативной памяти около 2 Гбайт).

Особый случай представляют собой ноутбуки, которые вы, возможно, захотите перевести в спящий режим (однако у меня это получалось плохо). В таком случае вся оперативная память сохраняется в разделе подкачки и ноутбук переходит в спящий режим. Для этого, разумеется, необходимо, чтобы раздел подкачки был больше, чем вся оперативная память (например, в полтора раза).

Специфические требования предъявляются и к крупным серверным системам. Например, Oracle для работы со своим сервером базы данных (версия 10.2) при расчете размеров раздела подкачки рекомендует использовать различные коэффициенты — в зависимости от доступного объема RAM:

- до 2 Гбайт — коэффициент 2;
- 2–8 Гбайт — коэффициент 1;
- более 8 Гбайт — коэффициент 0,75.

В 32-битных системах максимальный размер раздела подкачки составляет 2 Гбайт. Если вам требуется больше виртуальной памяти, можно использовать и несколько разделов подкачки. В таком случае гораздо целесообразнее перейти к работе с 64-битным дистрибутивом.

Время от времени возникает вопрос: можно ли (а, возможно, и следует ли) отказаться от раздела подкачки, если на компьютере установлен очень большой объем оперативной памяти. В принципе Linux может работать и без виртуальной памяти, но есть и важный аргумент в пользу создания раздела подкачки: если какая-то программа выйдет из-под контроля или по каким-то причинам потребует больше памяти, чем ожидалось, доступная память когда-нибудь будет израсходована. Это может привести к аварийному завершению следующего процесса, на который потребуется дополнительная память. Это может быть любой процесс — необязательно программа, вышедшая из-под контроля.

Строго говоря, раздел подкачки не решает этой проблемы, ведь и виртуальная память когда-нибудь закончится. Но из-за нарастающей загруженности виртуальной памяти программы в таком случае будут работать все медленнее и медленнее и вы заблаговременно поймете, что с компьютером что-то неладно. Прежде чем система откажет, вы успеете завершить программу, вызвавшую ошибку, командой `kill`. Если вы желаете подробнее изучить эту тему, обратитесь к одному из следующих сайтов:

- http://www.thomashertweck.de/linuxram.html;
- http://kerneltrap.org/node/3202.

Создание нового раздела подкачки. Если вы создали слишком маленький раздел подкачки или вам по каким-либо причинам понадобится еще один, создайте новый раздел (например, командой `fdisk`). В качестве типа диска укажите Linux swap (`82` в `fdisk`). Отформатировав раздел с помощью команды `mkswap`, вы можете активировать его командой `swapon`. Если все получится, дополните файл `etc/fstab`.

Чтобы не замедлять работу системы, старайтесь создавать на жестком диске только один раздел подкачки. В идеальном случае такой раздел должен находиться на малоиспользуемом или неиспользуемом диске.

Файлы подкачки. Чтобы не использовать раздел подкачки, вы можете выгружать память в отдельный файл подкачки. Конечно, это решение на крайний случай, так как из-за такой операции замедляется доступ к файловой системе. Единственное достоинство файла подкачки заключается в том, что для него не требуется отдельный раздел.

Обычно файлы подкачки сохраняются в разделе `/dev`. В первую очередь создаем командой `dd` пустой файл заданного размера. При этом в качестве источника данных используется `/dev/zero`. Из этого устройства можно считать сколько угодно нулевых байт. Размер нужно указывать в блоках, причем каждый блок равен 1024 байт. Кроме того, файл подкачки, подобно разделу подкачки, форматируется командой `mkswap` и активируется с помощью `swapon`.

В следующем примере мы создадим маленький файл подкачки размером около 1 Мбайт:

```
root#  dd bs=1024 if=/dev/zero of=/swapfile count=1000
1000+0 records in
1000+0 records out
root#  mkswap /swapfile 1000
Setting up swapspace, size = 1019904 bytes
root#  sync
root#  swapon -v /swapfile
swapon on device /swapfile
```

Файлы подкачки можно указать в `fstab`, как и разделы подкачки:

```
# Дополнение в /etc/fstab
/swapfile  none    swap   sw   0 0
```

Учитывайте также, что файлы подкачки не поддерживаются в файловой системе `btrfs`!

14.14. RAID

RAID — избыточный массив независимых жестких дисков. В этом разделе мы рассмотрим администрирование программного RAID-массива для Linux на базе пакета mdadm. Ради экономии места здесь будут описаны только уровни RAID-1 и RAID-0.

Обратите внимание — в Интернете есть много устаревших руководств по RAID. В них описывается конфигурация, основанная на raidtools (этот инструментарий в современных дистрибутивах Linux уже не используется).

Если вы собираетесь работать с новой файловой системой `btrfs` и хотите использовать массивы RAID-0, RAID-1 или RAID-10, вы можете отказаться от описанных здесь функций RAID, обеспечиваемых *драйверами для множественных устройств* (Multi Device Drivers). Дело в том, что функции RAID встроены в саму файловую систему `btrfs`.

Основы

Пакет mdadm. Если вы создали группу дисков RAID уже в ходе инсталляции, установите пакет mdadm. В нем содержится одноименная команда для администрирования RAID.

Рекомендуется пользоваться mdadm и при установке почтового сервера (MTA — агент пересылки сообщений), чтобы в случае проблем с RAID система сообщила об этом администратору по электронной почте. Если вы еще не занимались почтовым сервером, пока не устанавливайте такой агент. Вместо этого (в Debian и Ubuntu) при инсталляции укажите с помощью apt-get параметр --no-install-recommends.

Md_mod. Внутри Linux за работу программного RAID отвечает драйвер поддержки многодисковых устройств (Multi Devices Driver Support). В некоторых дистрибутивах этот драйвер интегрирован прямо в ядро, в других случаях при запуске системы автоматически загружается модуль md_mod (раньше он назывался просто md). В любом случае в dmesg должны содержаться соответствующие сообщения. Убедитесь, что существует псевдофайл /proc/mdstat. В нем хранится информация о текущем состоянии системы RAID.

Модуль md_mod создает логический уровень между драйвером, отвечающим за доступ к жесткому диску, и драйвером файловой системы (например, ext4). Этот модуль создает на нескольких разделах диска новое логическое устройство, к которому может обращаться драйвер файловой системы (/dev/mdn). После завершения конфигурации RAID вы используете не определенный раздел диска, а раздел /dev/mdn, где создаете свою файловую систему.

Mdadm.conf. Основным конфигурационным файлом RAID является /etc/mdadm/mdadm.conf. в этом файле, кроме некоторых глобальных настроек RAID, должны содержаться данные обо всех активных группах дисков RAID. Можно также с помощью /usr/share/mdadm/mkconf создать полностью новый конфигурационный файл. Это целесообразно делать в тех случаях, когда такой файл потерялся либо вы работаете с «живым» или восстановительным диском.

При такой конфигурации используется несколько необычный метод. Сначала, выполнив команду mdadm, вы создаете желаемые группы RAID или модифицируете их. Кроме того, вы дополняете имеющийся файл mdadm.conf в соответствии с действующей конфигурацией. Чтобы узнать основные показатели действующих групп RAID, выполните команду mdadm --examine -scan, а потом вставьте эти показатели с помощью >> в имеющийся конфигурационный файл.

```
root# mdadm --examine --scan >> /etc/mdadm/mdadm.conf
```

Если в файле mdadm.conf уже содержатся определенные ранее группы RAID, удалите эти сведения с помощью текстового редактора, чтобы какая-нибудь группа RAID не оказалась определена дважды. В следующих строках показано, как может быть построен файл mdadm.conf.

```
# Файл /etc/mdadm/mdadm.conf
DEVICE partitions
CREATE owner=root group=disk mode=0660 auto=yes
HOMEHOST <system>
```

```
MAILADDR root
ARRAY /dev/md0 level=raid1 num-devices=2 UUID=36c426b0:...
ARRAY /dev/md1 level=raid1 num-devices=2 UUID=71dfc474:...
ARRAY /dev/md2 level=raid1 num-devices=2 UUID=e0f65ea0:...
```

Статус. Актуальная информация о статусе RAID содержится в упоминавшемся выше файле `/proc/mdstat`. В следующем примере мы имеем три группы RAID-1, каждая из которых состоит из двух разделов. Все три группы активны и работают без ошибок: `[UU]` означает, что первый и второй разделы находятся в состоянии `up` (то есть без проблем).

```
root# cat /proc/mdstat
Personalities : [raid0] [raid1] [linear] [multipath]
                [raid6] [raid5] [raid4] [raid10]
md0 : active raid1 sda1[0] sdb1[1]
      979840 blocks [2/2] [UU]
md1 : active raid1 sda2[0] sdb2[1]
      1951808 blocks [2/2] [UU]
md2 : active raid1 sda3[0] sdb3[1]
      387730624 blocks [2/2] [UU]
unused devices: <none>
```

Разумеется, было бы неудобно постоянно заглядывать в этот файл и проверять, все ли в порядке. Гораздо лучше перепоручить эту задачу программе `mdadm --monitor`. Обычно она запускается сценарием Init-V `/etc/init.d/mdadm`. В зависимости от дистрибутива, может потребоваться сначала соответствующим образом сконфигурировать пакет `mdadm`. В Ubuntu, например, для этого нужно выполнить следующую команду:

```
root# dpkg-reconfigure mdadm
```

В процессе конфигурации `mdadm` система отобразит четыре диалоговых окна. На первом этапе можно активизировать автоматическую проверку избыточности — один раз в месяц (в 1:06 в первое воскресенье месяца данные разделов RAID сравниваются друг с другом). Такой контроль помогает заблаговременно обнаружить ошибки в тех сегментах или файлах, которые уже давно не читались и не изменялись. Внутри системы проверка избыточности осуществляется с помощью команды `checkarray`, запускаемой сценарием Cron `/etc/cron.d/mdadm`.

На втором и третьем этапах вы активизируете наблюдение за состоянием RAID и указываете, на какой электронный адрес посылать предупреждения либо сообщения о возникновении ошибок. Внутри системы такое наблюдение осуществляется с помощью команды `mdadm --monitor`. Она выполняется при запуске системы с помощью `/etc/init.d/mdadm`, если в `/etc/default/mdadm` есть настройка `START_DAEMON=true`. Электронный адрес сохраняется в файле `/etc/mdadm/mdadm.conf`. Если возникнет проблема, программа `mdadm` отошлет сообщение с уведомлением на адрес администратора. Чтобы этот механизм работал, на компьютере необходимо установить почтовый агент (MTA)! Нужный электронный адрес вы можете настроить в файле `/etc/mdadm/mdadm.conf` с помощью переменной `MAILADDR`.

На четвертом этапе конфигурации вы, наконец, указываете, должен ли сервер при перегрузке запускаться также в том случае, если он обнаруживает в разделе

RAID ошибку. Запускаться в таком случае должны, прежде всего, корневые серверы.

GRUB и RAID. Версия GRUB 0.97 совместима только с RAID-1. Если вы работаете с GRUB 0.97, но для системного раздела выбираете другой уровень RAID, вам потребуется отдельный загрузочный раздел. С GRUB 2 ситуация обстоит лучше: если в `grub.cfg` загружен модуль raid, то GRUB может считать данные ядра и файла `Initrd` прямо из группы RAID и отдельного загрузочного раздела для этого не требуется.

Чтобы компьютер, имеющий на диске системный раздел RAID, загружался и в том случае, когда жесткий диск неисправен, GRUB нужно установить в загрузочном секторе каждого жесткого диска. При работе с GRUB 2 для этого просто выполняется команда `grub-install /dev/mdn`. В GRUB 0.97 установка на каждом диске производится вручную.

Администрирование

Создание группы RAID-0. Чтобы создать группу RAID-0, вам потребуется как минимум два неиспользуемых раздела. Лучше, если эти разделы будут одинаковыми по размеру, но это необязательно. В зависимости от уровня RAID, при неравном размере разделов скорость работы может быть не оптимальной или же части большого раздела могут оставаться незаполненными.

Выбранные разделы должны быть обозначены как разделы RAID. Если для секционирования диска использовалась программа `fdisk`, то для идентификационного номера раздела необходимо установить шестнадцатеричное значение `fd`, применяя команду `T`. При работе с программой `parted` выполняется команда `set` *`номер_раздела`* `raid on`.

Далее разделы с названиями устройств `/dev/sda3` и `/dev/sdc1` объединяются в систему RAID-0. Форматировать эти разделы не требуется. Команда `fdisk -l` отображает примерную конфигурацию:

```
root# fdisk -l /dev/sda /dev/sdc
Disk /dev/sda: 320.0 GB, 320072933376 bytes
  Device Boot    Start   End    Blocks   Id    System
/dev/sda1            1   973   7815591   83    Linux
/dev/sda2          974  1034    489982+  82    Linux swap / Solaris
/dev/sda3         1035  2251   9775552+  fd    Linux raid autodetect
Disk /dev/sdc: 320.0 GB, 320072933376 bytes
  Device Boot    Start   End    Blocks   Id    System
/dev/sdc1            1  1217   9775521   fd    Linux raid autodetect
```

Всего одной команды `mdadm` достаточно для того, чтобы создать группу RAID-0 из двух разделов — `/dev/sda3` и `/dev/sdc1`.

```
root# mdadm --create /dev/md0 --level=0 --raid-devices=2 /dev/sda3 /dev/sdc1
mdadm: array /dev/md0 started.
```

Затем требуется поместить файловую систему в новый виртуальный раздел `/dev/md0`. Его можно подключить к файловой системе Linux командой `mount`. Раздел запрашивается через каталог `/striped`, но, разумеется, вы можете его переименовать.

```
root# mkfs.ext4 /dev/md0
root# mkdir /striped
root# mount /dev/md0 /striped/
```

Если все получится, нужно занести информацию о новом разделе в файл /etc/fstab. Во всех новых дистрибутивах Linux система RAID автоматически инициализируется процессом Init-V при перезапуске.

```
# in /etc/fstab
/dev/md0    /striped   ext4   defaults   0  0
```

Кроме того, необходимо дополнить конфигурационный файл одной строкой mdadm.conf, описывающей новую группу RAID-0. Команда mdadm --examine --scan возвращает строку в синтаксически правильном виде.

Создание группы RAID-1. Группа RAID-1 создается тем же способом, что и RAID-0. Только команда для создания системы RAID выглядит несколько иначе и содержит --level=1 вместо --level=0:

```
root# mdadm --create /dev/md0 --level=1 --raid-devices=2 /dev/sda3 /dev/sdc1
mdadm: array /dev/md0 started.
root# mkfs.ext4 /dev/md0
```

Если вы уже создали в /dev/md0 раздел RAID-0, нужно отключить раздел от дерева каталогов и деактивировать командой mdadm -stop. Только после этого вы сможете выполнить команду mdadm --create. Тем не менее, mdadm распознает, что разделы /dev/sda3 и /dev/sdc1 были использованы, и потребует подтвердить, что вы действительно хотите заново настроить /dev/md0.

Тестирование группы RAID-1. Чтобы протестировать, как действует группа RAID-1, еще до того как вы сохраните там важные данные, пометьте этот раздел как неисправный:

```
root# mdadm /dev/md0 --fail /dev/sdc1
```

Если при старте системы была запущена команда mdadm --monitor, то администратор, работающий на локальном компьютере, сразу же получит по электронной почте соответствующее уведомление. После этого вы сможете работать с группой, как и раньше, но теперь все изменения будут сохраняться на свободном пространстве, оставшемся на диске. Теперь /proc/mdstat отображает статус U_. Это говорит о том, что один раздел работает (U означает up), а один отсутствует (_).

```
root# cat /proc/mdstat
md0 : active raid1 sda3[1]
      979840 blocks [2/1] [_U]
```

Чтобы снова добавить /dev/sdc1 к /dev/md0, нужно специально удалить его как неисправный.

```
root# mdadm --remove  /dev/md0 /dev/sdc1
root# mdadm --add     /dev/md0 /dev/sdc1
```

Теперь автоматически начнется повторная синхронизация обоих разделов, на что, в зависимости от размера разделов, может понадобиться достаточно много времени (ориентировочно около 20 минут на 100 Гбайт). Правда, в ходе синхрони-

зации вы можете продолжать работу. Файловая система будет функционировать несколько медленнее.

```
root# cat /proc/mdstat
md0 : active raid1 sda3[1] sdc1[2]
      485454656 blocks [2/1] [U_]
      [>.....................] recovery = 3.0% (14577856/485454656)
                              finish=72.8min speed=107724K/sec
root# mdadm --detail /dev/md0   (Пока синхронизация продолжается)
...
          State : clean, degraded, recovering
 Active Devices : 1
Working Devices : 2
 Failed Devices : 0
  Spare Devices : 1
 Rebuild Status : 75% complete
...
  Number    Major    Minor    RaidDevice    State
    0         3        3         0          active sync       /dev/sda3
    1         0        0         -          removed
    2        22        2         1          spare rebuilding  /dev/sdc1
root# mdadm --detail /dev/md0  (После завершения синхронизации)
...
          State : clean
 Active Devices : 2
Working Devices : 2
 Failed Devices : 0
  Spare Devices : 0
...
  Number    Major    Minor    RaidDevice    State
    0         3        3         0          active sync   /dev/sda3
    1        22        2         1          active sync   /dev/sdc1
```

Замена неисправного жесткого диска из RAID-1. К счастью, это бывает редко, но если на жестком диске действительно появляются неисправности и `mdadm` пометит отдельные разделы как дефектные, все разделы этого диска необходимо специально удалить из соответствующих групп RAID.

```
root# mdadm --remove /dev/md0 /dev/sdc1
...
```

Кроме того, вам нужно как можно скорее заменить диск. На новом жестком диске должно быть достаточно места, чтобы создать разделы такого же размера, как и на имеющихся жестких дисках.

Будьте внимательны — из группы нужно извлечь именно неисправный диск, а не рабочий! Этот совет кажется банальным, но если на компьютере два или более конструктивно одинаковых диска, найти нужный диск не так просто, как кажется. Уникальным признаком жесткого диска является только его серийный номер! Чтобы узнать, какой серийный номер соответствует названию устройства, используйте команды `hdparm` или `smartctl`. Для выполнения обеих команд необходимо установить одноименные пакеты.

```
root# smartctl -i /dev/sdc
...
Device Model:  SAMSUNG HD403LJ
Serial Number: S0NFJ1MPA07356

root# hdparm -i /dev/sdc
/dev/sdb:
...
Model=SAMSUNG HD403LJ, FwRev=CT100-12,
SerialNo=S0NFJ1MPA07356
```

После замены жесткого диска вам следует создать на новом диске разделы, которые будут не меньше, чем уже имеющиеся разделы RAID. При этом вам очень пригодится команда `sfdisk` (см. подраздел «Программа sfdisk (MBR)» раздела 14.3). Разделы необходимо пометить как относящиеся к RAID (шестнадцатеричный идентификационный код `fd`). Когда эта подготовительная работа будет завершена, останется добавить разделы нового жесткого диска к группам RAID:

```
root# mdadm --add /dev/md0 /dev/sdc1
...
```

Теперь ядро начнет синхронизировать разделы нового жесткого диска с имеющимися данными RAID. Статус синхронизации отслеживается с помощью команды `cat /proc/mdstat`.

СОВЕТ

Настоятельно рекомендую вам поучиться исправлять RAID на тестовой системе, никуда не торопясь. Дефект жесткого диска можно имитировать, пометив диск, как неисправный, командой mdadm --fail или на время отключив диск от сети (конечно же, не на ходу!).

Деактивация группы RAID. Команда `mdadm --stop` деактивирует группу RAID. Предварительно нужно отключить от дерева каталогов файловую систему этой группы с помощью команды `umount`.

```
root# umount /mount-directory/
root# mdadm --stop /dev/md0
```

Повторная активация группы RAID. Теперь, если после выполнения `mdadm --stop` вы не внесли в разделы группы никаких изменений, можно снова собрать и активировать группу RAID командой `mdadm --assemble` — потери данных исключаются.

```
root# mdadm --assemble /dev/md0 /dev/sda3 /dev/sdc1
mdadm: /dev/md0 has been started with 2 drives.
```

Анализ разделов. Во всех разделах жесткого диска, которые вы объединили в группы RAID с помощью `mdadm`, в специальных блоках сохраняется контекстная информация (метаданные). Эту информацию можно считать с помощью команды `mdadm --query`, например, чтобы узнать статус неизвестной системы.

```
root# mdadm --query /dev/sda3
/dev/sda3: is not an md array
```

```
/dev/sda3: device 0 in 2 device active raid1 md0. Use mdadm --examine
  for more detail.
root# mdadm --query /dev/md0
/dev/md0: 9.32GiB raid1 2 devices, 0 spares. Use mdadm --detail for more detail.
/dev/md0: No md super block found, not an md component.
```

Команда `mdadm --examine` возвращает подробную информацию о разделе, входящем в группу RAID:

```
root# mdadm --examine /dev/sda3
/dev/sda3:
     Raid Level : raid1
   Raid Devices : 2
  Total Devices : 2
Preferred Minor : 0
    Update Time : Thu Nov 9 17:02:39 2006
          State : clean
 Active Devices : 2
Working Devices : 2
  Number    Major    Minor    RaidDevice    State
0   0         3        3          0         active sync   /dev/sda3
1   1        22        1          1         active sync   /dev/sdc1
```

Аналогично `mdadm --detail` выдает подробную информацию о группе RAID:

```
root# mdadm --detail /dev/md0
/dev/md0:
      Version : 00.90.03
Creation Time : Thu Nov 9 16:55:35 2006
   Raid Level : raid1
   Array Size : 9775424 (9.32 GiB 10.01 GB)
  Device Size : 9775424 (9.32 GiB 10.01 GB)
 Raid Devices : 2
Total Devices : 2
...
```

Контроль целостности данных. Как убедиться, что все избыточно сохраненные данные действительно корректны? Как правило, RAID-система проводит тестирование целостности данных лишь в случае считывания или записи данных. Но многие данные остаются неизменными в течение целых месяцев. Итак, чтобы удостовериться, что все жесткие диски в порядке, система RAID должна считать все блоки с данными и сравнить избыточную информацию. Этот процесс также называется патрулированием или контролем целостности памяти (scrubbing).

```
root# echo check > /sys/block/mdn/md<n>/sync_action
```

Если при этом будут выявлены ошибки, то их можно исправить:

```
root# echo check > /sys/block/mdn/md<n>/sync_action
```

В Debian и Ubuntu эта задача выполняется сценарием `/usr/share/mdadm/checkarray`, который ежемесячно запускается программой `cron`. В Fedora эту роль играет cron-сценарий `/etc/cron.weekly/99-raid-check`.

Удаление метаданных RAID. Как правило, бывает полезно сохранять метаданные RAID в неиспользуемых секторах раздела. Однако если позже вы пожелаете задействовать этот жесткий диск для других целей, метаданные RAID могут представлять проблему: установочные программы Linux и `mdadm` распознают остатки конфигурации RAID и не поймут, что теперь эти разделы нужно использовать иначе.

Вам пригодится следующая команда, которую необходимо применить ко всем разделам RAID:

```
root# mdadm --zero-superblock /dev/sda3
```

Если вы пробовали работать с BIOS-RAID, можете удалить соответствующие метаданные на всех жестких дисках с помощью команды `dmraid -r -E`.

14.15. Менеджер логических томов (LVM)

Менеджер логических томов (Logical Volume Manager, LVM) — это логический слой, расположенный между файловой системой и разделами диска. Принцип, приоритеты и порой совершенно неясная номенклатура LVM здесь рассматриваться не будут. Поговорим об администрировании LVM.

Помощь при конфигурации. В некоторых дистрибутивах предоставляются инструменты для администрирования LVM, не требующие остановки системы. В Fedora и Red Hat при конфигурации используется `system-config-lvm`, в SUSE — модуль YaST Система ▸ LVM.

Хотя эти программы и помогают при конфигурации, для работы с ними требуется глубоко понимать концепции, связанные с LVM. Обратите внимание, что при изменении размера, как правило, увеличиваются или уменьшаются только сами логические тома, а не содержащиеся в них файловые системы. Размер файловой системы необходимо изменять *до* уменьшения и *после* увеличения соответствующих логических томов.

Модуль dm_mod. Внутри системы за управление LVM отвечает модуль ядра `dm_mod`. В некоторых дистрибутивах функции LVM скомпилированы прямо внутри ядра, поэтому не отображаются при выводе результата команды `lsmod`.

GRUB. Если LVM создан уже при установке системы, то системный раздел также может находиться в логическом томе. В любом случае с LVM совместима только версия GRUB 2. Если вы работаете с GRUB 0.97, вам для этого потребуется отдельный загрузочный раздел, без LVM.

RAID. Можно комбинировать LVM и RAID. Обычно для этого создается группа RAID, а затем полученное устройство `/dev/mdn` используется в качестве физического тома (PV).

Особый случай представляет собой RAID-0. Этот вариант RAID поддерживается непосредственно LVM. Чтобы пользоваться этой функцией, вам потребуется создать на одном или нескольких жестких дисках по физическому тому. Эти тома объединяются в группу томов (Volume Group). Теперь с помощью команды `lvcreate -i n` вы можете создать логический том, данные которого будут разделены между несколькими (*n*) физическими томами.

Команды LVM. Для администрирования LVM применяется целый спектр команд. Их названия начинаются с `pv`, `vg` или `lv`, в зависимости от того, для работы с чем они предназначены — с *физическими томами* (`pv`), *группами томов* (`vg`) или *логическими томами* (`lv`). Важнейшие команды перечислены в табл. 14.10. Они входят в состав пакета `lvm2`, который сначала нужно установить.

Таблица 14.10. Обзор команд для работы с LVM

Команда	Функция
lvcreate	Создает в группе томов новый логический том
lvdisplay	Сообщает подробную информацию по определенному логическому тому
lvextend	Увеличивает логический том
lvreduce	Уменьшает логический том
lvremove	Удаляет логический том
lvrename	Переименовывает логический том
lvscan	Перечисляет логические тома
pvcreate	Обозначает раздел или устройство как физический том
pvdisplay	Сообщает подробную информацию по физическому тому
pvremove	Удаляет обозначение «физический том» неиспользуемого физического тома
pvscan	Перечисляет физические тома
vgchange	Изменяет атрибуты группы томов
vgcreate	Создает новую группу томов из одного или нескольких физических томов
vgdisplay	Сообщает подробную информацию о группе томов
vgextend	Увеличивает группу томов на один том
vgmerge	Объединяет две группы томов
vgreduce	Уменьшает группу томов, убирая из нее неиспользуемый том
vgrename	Переименовывает группу томов
vgscan	Перечисляет все группы томов

Можно работать не с отдельными командами, а выполнять все администрирование LVM с помощью команды `lvm`, в качестве первого параметра которой вы указываете желаемую операцию. Таким образом, команды `lvcreate` и `lvm lvcreate` равнозначны.

Примеры. В следующих примерах показано, как применять некоторые команды LVM. При этом предполагается, что при установке LVM не создавался. Теперь дополнительный жесткий диск `/dev/sdc` должен использоваться через LVM. Секционирование диска выглядит так:

```
root# fdisk -l /dev/sdc
Disk /dev/sdc: 320.0 GB, 320072933376 bytes
  Device Boot    Start    End     Blocks    Id    System
  /dev/sdc1          1   1217    9775521    8e    Linux LVM
  /dev/sdc2       1218   2434    9775552+   8e    Linux LVM
```

Чтобы инициализировать LVM, выполните команды `modprobe` и `vgscan`. Как только система LVM будет создана, модуль ядра LVM автоматически выполнится при запуске компьютера. Иначе говоря, систему требуется инициализировать вручную только в первый раз:

```
root# modprobe dm_mod
root# vgscan
  Считывание всех физических томов (подождите, пожалуйста...)
  Группы томов не найдены
```

По методическим соображениям мы сначала создадим LVM в разделе /dev/sdc1, а потом добавим к нему /dev/sdc2. Если и так ясно, что вы собираетесь использовать для LVM весь жесткий диск, гораздо проще обозначить сам диск или как минимум его раздел максимального размера с помощью команды pvcreate как предназначенный для LVM.

```
root# pvcreate /dev/sdc1
  Физический том "/dev/sdc1" успешно создан
```

Теперь нужно объединить все физические тома в группу. В данном примере у нас сначала есть только один физический том, но выполнить такой шаг необходимо. Команде vgcreate также нужно сообщить желаемое название группы томов. В этом примере группа томов получает название myvg1:

```
root# vgcreate myvg1 /dev/sdc1
  Группа томов "myvg1" успешно создана
```

Теперь myvg1 представляет собой своего рода пул данных, который, правда, пока не используется. Для его применения вам нужно будет создать в myvg1 логический том, то есть своего рода виртуальный раздел. При этом необходимо передать команде lvcreate три параметра: желаемый размер и название логического тома, а также название существующей группы томов:

```
root# lvcreate -L 2G -n myvol1 myvg1
  Создан логический том "lvol0"
```

Одновременно команда создает файл /dev/myvg1/myvol1. При этом используется ссылка на файл /dev/mapper/myvg1-myvol1. Теперь логический том можно применять под любым из обоих названий устройств как обычный раздел жесткого диска.

Чтобы создать файловую систему в *логическом томе*, используйте, например, команду mkfs.ext4 или mkfs.xfs:

```
root# mkfs.ext4 /dev/myvg1/myvol1
```

С помощью команды mount можете проверить, все ли получилось:

```
root# mkdir /test
root# mount /dev/myvg1/myvol1 /test
```

Увеличение файловой системы. Причина, по которой стоит пользоваться логическими томами, заключается в том, что появляется возможность постепенно увеличивать файловую систему, не секционируя диск заново. В следующем примере созданная выше файловая система (dev/myvg1/myvol1 посредством /test) увеличивается с 2 до 3 Гбайт. Команда df позволяет узнать мощность /test перед внесением изменений:

```
root# df -h -T /test
Filesystem   Typ   Size   Used   Available   Used%   Mounted on
```

```
/dev/mapper/myvg1-myvol1
            ext4  2,0G   760M        1,2G     40%   /test
```

Для этого сначала нужно увеличить логический том. Чтобы это сделать, сообщим название устройства и новый размер команде `lvextend`. Дополнительно следует соответствующим образом увеличить и файловую систему ext4.

```
root# lvextend -L 3G /dev/myvg1/myvol1
  Увеличение логического тома myvol1 до 3,00 GB
  Размер логического тома myvol1 успешно изменен
root# resize2fs /dev/myvg1/myvol1
```

Команда `df` доказывает, что все сработало:

```
root# df -h -T /test
File system   Typ   Size   Used   Available   Used%   Mounted on
/dev/mapper/myvg1-myvol1
              ext4  3,0G   760M   2,1G          27%   /test
```

В принципе логический том можно и уменьшить. Для этого нужно сначала обязательно отключить ту или иную файловую систему от дерева каталогов, проверить с помощью `fsck.ext4` и, наконец, уменьшить командой `resize2fs`. Только теперь можно уменьшить лежащий в основе логический том командой `lvreduce`.

Пока в пуле памяти *(группы томов)* еще есть место, виртуальные разделы (логические тома) можно с легкостью увеличить. Но что делать, если группа томов заполнена? В таком случае на любом жестком диске вашего компьютера нужно создать новый раздел, пометить этот раздел как *физический том* и добавить его в группу томов командой `vgextend`.

На примере двух следующих команд показано, как этот механизм работает с разделом `/dev/sdc2`. Таким образом, общий объем `myvg1` достигает около 19 Гбайт, из которых почти 16 Гбайт свободно:

```
root# pvcreate /dev/sdc2
  Физический том "/dev/sdc2" успешно создан
root# vgextend myvg1 /dev/sdc2
  Группа томов "myvg1" успешно расширена
root# vgdisplay myvg1
...
  Размер группы томов         18,64 GB
  Распределено PE / Размер    640 / 2,50 GB
  Свободно PE / Размер        4132 / 16,14 GB
...
```

Мгновенные снимки. Менеджер логических томов позволяет создавать мгновенные снимки. Мгновенный снимок — это неизменяемая копия файловой системы, отражающая ее состояние на определенный момент времени. Мгновенный снимок можно интегрировать в дерево каталогов как отдельную файловую систему. Если та файловая система, с которой был сделан снимок, изменится, то исходные данные будут архивированы для мгновенного снимка. Уже при создании мгновенного снимка необходимо указать, сколько дискового пространства LVM

должен зарезервировать для такой архивации. Если это место будет израсходовано, то мгновенный снимок становится непригодным для дальнейшего использования.

Мгновенные снимки менеджера логических томов предлагают значительно меньше функций, чем мгновенные снимки `btrfs`. Как правило, мгновенные снимки LVM применяются при резервном копировании. Вы можете убедиться, что во время резервного копирования файлы не изменяются и что сама резервная копия является целостной.

Приведенные ниже команды демонстрируют, как мгновенный снимок логического тома сначала создается командой `myvol1`, потом подключается к дереву каталогов в каталоге `/media/backup`, затем из него делается резервная копия, мгновенный снимок удаляется из дерева каталогов и, наконец, удаляется. Пока длится резервное копирование, логический том `myvol1` можно сколько угодно использовать и далее — например, в качестве пространства для хранения информации из сервера базы данных. Но изменения, производимые во время резервного копирования, так или иначе, не должны превышать 100 Мбайт. (В ходе резервного копирования команда `lvdisplay /dev/vg1/snap` позволяет узнать, сколько процентов отведенного пространства уже используется.)

Обратите внимание на то, что имена устройств используемых логических томов требуется задавать в формате `/dev/vgname/lvname`, а не в формате `/dev/mapper/vgname-lvname`!

```
root# lvcreate -s -L 100M snap /dev/myvg1/myvol1
Создан логический том snap
root# mkdir /media/backup
root# mount /dev/vg1/snap /media/backup
root# backup-script /media/backup  (Создание резервной копии)
root# umount /media/backup
root# lvremove /dev/vg1/snap
```

14.16. SMART

Аббревиатура SMART означает Self-Monitoring, Analysis and Reporting Technology (Система слежения, анализа и отчетности). Данная система используется почти со всеми имеющимися на рынке жесткими дисками IDE, SATA и SCSI. Благодаря SMART на диске периодически сохраняются различные параметры, которые позволяют сделать вывод о потенциальных неисправностях жесткого диска, а также о том, сколько он еще проработает. Через специальный интерфейс параметры SMART можно считывать. Регулярное отслеживание этих параметров операционной системой — это своего рода механизм заблаговременного обнаружения проблем. С его помощью неисправности можно предупредить задолго до того, как они, возможно, вызовут потери данных.

В этом разделе сделан обзор инструментов, используемых в Linux для считывания параметров SMART. Более подробная информация сообщается в «Википедии», а также на сайтах http://sourceforge.net/apps/trac/smartmontools/wiki и http://www.linuxjournal.com/article/6983.

Условия. Для работы SMART есть некоторые предварительные условия.

- Диск должен поддерживать систему SMART. Чтобы определить, так ли это, выполните, например, команду `hdparm -I/dev/sdx`.
- Диск должен быть внутренним или иметь тип eSata. С внешними дисками, подключаемыми через USB или Firewire, функции SMART, к сожалению, не используются.
- Если жесткий диск управляется через аппаратный контроллер RAID, функции SMART могут применяться только в отдельных случаях (подробности в справке `man smartctl` по параметру `-d`).

Gnome. Современные версии Gnome автоматически отслеживают SMART-статус жестких дисков и выводят предупреждение в случае, если существует угроза дефекта диска. «За кулисами» за этот процесс отвечает программа `/usr/lib/gnome-disk-utility/gdunotification-daemon`, содержащаяся в пакете `gnome-disk-utility`. Программы Gnome Disks или Palimpsest позволяют, кроме того, отображать SMART-данные диска.

Программа smartctl. SMART-статус можно узнать и через командную строку, что особенно важно при работе с серверами. Соответствующая команда `smartctl` в большинстве дистрибутивов входит в состав пакета `smartmontools`, который зачастую требуется установить дополнительно. В некоторых дистрибутивах при этом автоматически инсталлируется почтовый сервер (MTA), позволяющий при необходимости рассылать уведомления SMART по электронной почте. На сервере такую систему иметь полезно, а на локальном компьютере, как правило, она не нужна. В Debian и Ubuntu можно отменить установку почтового сервера, если сообщить команде `apt-get` параметр `--no-install-recommends`.

В простейшей форме команда `smartctl` выдает различную статусную информацию. Если `smartctl -i` выводит в последней строке сообщение SMART support is Disabled (Поддержка SMART отключена), активируйте SMART командой `smartctl -s on`.

```
root# smartctl -i /dev/sdb
Версия smartctl 5.41 ...,Copyright (C) 2002-11 Bruce Allen
Модель устройства:       SAMSUNG SSD 830 Series
Серийный номер:          S0Z3NYAC210778
Идентификационный номер устройства LU WWN   5 002538 043584d30
Версия прошивки:         CXM03B1Q
Пользовательская мощность:  128.035.676.160 bytes [128 Гб]
Размер сектора:          512 байт логический/физический
Устройство:              не входит в базу данных smartctl [подробности: -P showall]
Версия ATA:              8
Стандарт ATA:            ACS-2 версия 2
Местное время:           Ср. Июл 4 09:45:45 2012 среднеевропейское время
Поддержка SMART:         доступна – устройство поддерживает SMART.
Поддержка SMART:         включена
```

Команда `smartctl -H`, или `smartctl --health`, указывает, нормально ли работает жесткий диск и будет ли он работать через 24 часа. Если в качестве результата `smartctl` в данном случае не выдаст `PASSED`, нужно *немедленно* начать полное резервное копирование!

```
root# smartctl -H /dev/sda
...
SMART overall-health self-assessment test result: PASSED
```

Команда `smartctl -A` (или `smartctl -attributes`) выдает полный список специфичных заводских атрибутов жесткого диска. Для этих атрибутов не существует строгого стандарта, но важнейшие из них поддерживаются многими производителями жестких дисков. При интерпретации значений очень важны два столбца: `VALUE` указывает актуальное значение, а `THRESH` — предельную величину. Если актуальная величина превышает предельную, ждите проблем — жесткий диск свое уже отработал.

Базовым нормальным значением для большинства показателей является `100`. Например, значение `Power_On_Hour` у нового жесткого диска равно `100`. По истечении определенного периода эксплуатации (в часах) значение снижается до `99` и т. д. Чтобы узнать, сколько часов уже эксплуатировался жесткий диск, посмотрите значение в столбце `RAW_VALUE`. У тестового диска это `3360`, что равно около 420 рабочих дней по 8 часов. На некоторых жестких дисках срок эксплуатации измеряется в минутах или секундах. В таком случае, чтобы узнать верное значение, укажите `-v 9,minutes` или `-v 9,seconds` соответственно.

Следующий, несколько сокращенный, вывод взят из оценки состояния жесткого диска SATA, проработавшего около девяти месяцев. Признаков проблем нет.

```
root# smartctl -A /dev/sda
...
Vendor Specific SMART Attributes with Thresholds:
ID#  ATTRIBUTE_NAME          VALUE  WORST  THRESH  TYPE      UPDATED  RAW_VALUE
  1  Raw_Read_Error_Rate     109    099    006     Pre-fail  Always   24386832
  3  Spin_Up_Time            096    095    000     Pre-fail  Always   0
  4  Start_Stop_Count        100    100    020     Old_age   Always   167
  5  Reallocated_Sector_Ct   100    100    036     Pre-fail  Always   0
  7  Seek_Error_Rate         065    060    030     Pre-fail  Always   3391200
  9  Power_On_Hours          100    100    000     Old_age   Always   451
...
198  Offline_Uncorrectable 100    100    000     Old_age   Offline  0
199  UDMA_CRC_Error_Count  200    200    000     Old_age   Always   0
```

Команда `smartctl -l error` сообщает информацию о пяти последних возникших ошибках. Часто результат просто пуст (`No Errors Logged`). Единичные неповторяющиеся ошибки — это, как правило, не повод для беспокойства.

```
root# smartctl -l error /dev/sda
SMART Error Log Version: 1
No Errors Logged
```

Проведение самодиагностики. SMART предусматривает различные варианты самодиагностики системы, позволяющие еще точнее определить, в каком состоянии находится жесткий диск. Такие тесты начинаются с `smartctl -t short/long`. Краткий тест длится несколько минут, а подробный (`long`) в некоторых случаях может занять несколько часов. Тест проводится в фоновом режиме, то есть вы можете продолжать

работу. После того как тест будет закончен, его результаты можно отобразить командой `smartctl -l selftest`. В столбце `Remaining` указано, сколько еще времени будет выполняться тест. Если значение выше 0 %, то тест еще не окончен! Значение `LifeTime` указывает, сколько времени проработал диск; `LBA` определяет место (сектор), где возникла первая ошибка.

В следующем выводе показано выполнение трех тестов самодиагностики; первый был проведен почти сразу же после подключения диска (через 50 часов эксплуатации), два оставшихся — примерно через 2600 часов.

```
root# smartctl -t short /dev/sda
Num   Test_Description   Status                  Remaining   LifeTime   LBA
# 1   Extended offline   Completed without error    00%       2592     -
# 2   Short offline      Completed without error    00%       2591     -
# 3   Short offline      Completed without error    00%         40     -
```

Автоматическое наблюдение (smartd). Программа `smartctl` — это очень интересный инструмент, предназначенный для сбора данных о жестком диске. Однако для регулярного наблюдения за всеми дисками программа неудобна. При необходимости регулярного наблюдения аналогичную задачу выполняет программа `smartd`. Это демон (системная служба). Команды для автоматического запуска отличаются от дистрибутива к дистрибутиву.

Команда `smartd` управляется через `/etc/smartd.conf`. В некоторых дистрибутивах сценарий Init-V также интерпретирует файлы `/etc/sysconfig/smartmontools` или `/etc/default/smartmontools`. В этих файлах имеются дополнительные командные параметры для `smartd`. В Debian и Ubuntu потребуется внести в `smartmontools` настройку `start_smartd=yes`, иначе программа не запустится!

Обычная конфигурация компьютера с двумя жесткими дисками SATA (`/dev/sda` и `/dev/sdb`) выглядит следующим образом:

```
# Файл /etc/smartd.conf
/dev/sda -d sat -H -m root -M test
/dev/sdb -d sat -H -m root -M test
```

Это означает, что здоровье обоих жестких дисков будет проверяться каждые полчаса (как при использовании `smartctl -H`). Если при этом обнаружится ошибка, `smartd` отошлет по электронной почте сообщение локальному пользователю root (для этого вам в любом случае потребуется локальный почтовый сервер). Параметр `-d sat` обозначает жесткие диски как устройства SATA; `-M test` служит для того, чтобы узнать, работает ли в принципе рассылка электронной почты. Начните со `smartd`:

```
root# service smartd start
```

Если вы получите тестовое электронное сообщение, удалите из конфигурации `-M test`. В файле `smartd.conf`, входящем в комплект вместе с `smartmontools`, есть еще несколько примеров конфигурации.

Пакет smart-notifier. Вы можете получать визуальные сообщения о возникновении проблем, угрожающих жесткому диску. Для этого установите пакет `smart-notifier` и измените `smartd.conf` согласно следующему образцу:

```
# /etc/smartd.conf
DEVICESCAN -m root -M test -M exec /usr/share/smartmontools/smartd-runner
```

Теперь для сообщений SMART будет выполняться команда `smartd-runner`. Она пересылает уведомление `smartd` через систему связи D-Bus. Чтобы предупреждение `smartd` отображалось и на действующем Рабочем столе, в фоновом режиме должна запуститься программа `smart-notifier`. Это условие выполняется во всех локальных системах, интерпретирующих `/etc/xdg/autostart`.

Обратите внимание, что для параметра `-M exec` в `smartd.conf` также нужен параметр `-m`. Иными словами, вы не можете обойтись без электронного сообщения. Напротив, параметр `-M test` вы можете удалить после того, как убедитесь, что система уведомления функционирует.

14.17. SSD-TRIM

Linux без малейших проблем работает с SSD (твердотельными дисками). Разумеется, работа протекает гораздо быстрее, чем на традиционных жестких дисках. Правда, в одном аспекте твердотельные диски коренным образом отличаются от жестких.

Дело в том, что для внутрисистемной оптимизации ячеек памяти необходимо, чтобы операционная система сообщила твердотельному диску информацию следующего характера: какие блоки данных файловой системы в настоящее время не используются (например, из-за того, что в конкретном блоке памяти был удален файл). Этот процесс получил название SSD-TRIM. Базовая техническая информация по SSD-TRIM изложена в «Википедии»: http://ru.wikipedia.org/wiki/Твердотельный_накопитель.

Ни в одном из известных мне дистрибутивов Linux функция TRIM не выполняется по умолчанию. Такая стандартная настройка может быть объяснена тем, что снижение производительности, обусловленное отсутствием TRIM, на современных твердотельных дисках (при нормальном использовании) является незначительным. Кроме того, SSD-TRIM можно выполнять разными способами, и каждый из них имеет свои достоинства и недостатки.

Варианты TRIM. Профессионалы Linux, для которых непременным условием работы является оптимальная скорость функционирования твердотельного диска, должны сами решать вопросы, связанные с SSD-TRIM. Для таких экспертов лучше всего подходит один из двух вариантов: онлайн-TRIM, при котором Linux сразу же уведомляет твердотельный диск о каждом удаленном файле, и пакетный TRIM (batch TRIM), при котором операция проводится с регулярными промежутками (например, раз в неделю).

Недостаток первого варианта заключается в том, что внутрисистемные работы по наведению порядка на твердотельном диске производятся именно тогда, когда диск и так занят — выполняет интенсивные процессы записи. При онлайн-TRIM любой такой процесс записи замедляется.

Против пакетного TRIM говорит тот факт, что в назначенное время выполняются сразу все накопившиеся операции за целый день или за целую неделю. Во

время проведении этого процесса (на больших твердотельных дисках на это может уходить до нескольких минут, но обычно все происходит быстрее), каждая операция доступа к SSD протекает заметно медленнее.

СОВЕТ

Я рекомендую использовать второй вариант, то есть пакетный TRIM. Во-первых, его проще конфигурировать, а во-вторых, при одноразовом исполнении всех накопившихся TRIM-операций этот процесс не будет надолго замедлять текущую работу.

Онлайн-TRIM. Чтобы активизировать метод онлайн-TRIM, нужно добавить параметр `discard` в файле `/etc/fstab` тех файловых систем, в которых вы хотите внедрить такой способ. Этот параметр доступен при работе с файловыми системами ext4, btrfs и xfs. Следите за тем, чтобы в `/etc/fstab` не закрались какие-нибудь синтаксические ошибки! В списке `mount`-параметров не могут содержаться пробелы.

```
# Файл /etc/fstab
UUID=018e... / ext4 errors=remount-ro,user_xattr,discard 0 1
```

Пакетный TRIM. Чтобы вручную запустить пакетный TRIM, выполните в окне терминала команду `fstrim`:

```
root# fstrim -v /
/: 6493835264 байт были отсечены
```

Действие команды TRIM можно ограничивать и большими блоками данных. В таком случае SSD уведомляется лишь о наличии свободных блоков данных указанного размера и крупнее. В таком случае `fstrim` может выполняться значительно быстрее, и это, безусловно, хорошо.

```
root# fstrim -m 64K -v /
```

Чтобы автоматизировать этот процесс, создайте новый системный файл Cron (можно с параметром `-m`). Если вы разделили вашу файловую систему на много сегментов, то придется выполнить `fstrim` для каждого `mount`-пункта.

```
# Файл /etc/cron.weekly/fstrim
/sbin/fstrim /
```

Ограничения и особые случаи. Программный RAID не поддерживает операцию TRIM. Если вы шифруете вашу файловую систему (подробнее об этом — в следующем разделе), то, в принципе, совместное использование неиспользуемых блоков памяти можно организовать через TRIM. Но безопасность зашифрованных файлов при этом снижается. Напротив, при взаимодействии с менеджером логических томов TRIM никаких сложностей не доставляет.

Принципиально `mount`-параметр `discard` может использоваться и с разделом подкачки. В настольных системах этого не требуется, поскольку при каждом перезапуске компьютера Linux заново предоставляет весь раздел подкачки в совместное применение с помощью TRIM. Но на сервере, который может непрерывно работать в течение многих месяцев, этот параметр в определенных обстоятельствах бывает целесообразен.

14.18. Шифрование

Иногда ноутбуки или USB-флешки теряются или попадают в руки к ворам. В таком случае вы утрачиваете не только устройство, но и хранящиеся на нем данные, что вдвойне плохо. Чужие люди могут получить доступ к данным онлайн-банкинга, номерам страховки, историям болезни, коммерческим или военным тайнам. Этого нельзя допускать. Достаточно провести несложное шифрование файловой системы, чтобы надежно защитить свои данные. в этом разделе сообщается некоторая базовая информация по работе с зашифрованными файлами и файловыми системами.

Шифрование отдельных файлов

Gpg. Отдельный файл удобнее всего зашифровать командой `gpg`. Если вы введете `gpg -c`, система попросит вас дважды набрать пароль, затем зашифрует указанный файл и сохранит результат под именем `file.gpg`. При этом по умолчанию применяется алгоритм шифрования CAST5. Теперь исходный файл можно удалить. Команда `gpg -d` снова восстанавливает файл.

```
user$ gpg -c file
Введите пароль: **********
Повторите пароль: **********
user$ gpg -d файл.gpg > file
Введите пароль: **********
```

Команда `gpg` может использовать при кодировании или декодировании открытый или закрытый ключ, может ставить на файлах цифровые подписи, управлять ключами и т. д. Описание бесчисленных параметров в справке `man` занимает около 50 страниц! Но вручную команда `gpg` применяется редко. Чаще ее используют почтовые клиенты, чтобы (более или менее автоматически) ставить на электронные сообщения подписи или шифровать эти сообщения.

Шифрование файловой системы (USB-флешка, внешний жесткий диск)

Dm_crypt и LUKS. Уже давно разработаны многочисленные методы для шифрования файловых систем: CryptoFS, eCryptfs, Enc-FS, Loop-AES и LUKS. Некоторые из этих методов применяются до сих пор, другие — зачастую из-за недостаточной безопасности — уже заброшены. В настоящее время самым популярным методом является LUKS (Linux Unified Key Setup).

Метод LUKS основан на модуле ядра dm_crypt, который обогащает применяемый в LVM модуль отображения устройств функциями шифрования. Модуль dm_crypt — это логический слой, расположенный между зашифрованными исходными данными жесткого диска и файловой системой, которую видит пользователь. Модуль поддерживает различные алгоритмы шифрования, его можно комбинировать с LVM — часто так и делается, но это совсем не обязательно.

Модуль dm_crypt можно использовать и в такой системе, которая не работает с LVM!

LUKS добавляет к зашифрованным данным заголовок, в котором содержится метаинформация. В заголовке среди прочего указано, каким методом были зашифрованы данные. LUKS очень упрощает интеграцию зашифрованных носителей данных в Linux.

Cryptsetup. Чтобы создавать зашифрованные файловые системы, воспользуйтесь командой `cryptsetup` из одноименного пакета. В следующих строках показано, как сначала отформатировать USB-флешку (`/dev/sdh1`) как криптоустройство (`luksFormat`), а потом активизировать устройство под произвольно выбранным именем `mycontainer` (`luksOpen`). Разумеется, степень защиты ваших данных прямо пропорциональна сложности пароля, который может состоять и из нескольких слов. Рекомендуется, чтобы в пароле было минимум 20 символов.

Кроме того, вы можете использовать `/dev/mapper/mycontainer` как раздел жесткого диска или логический том, то есть создать файловую систему, подключить ее к дереву каталогов и т. д. После выполнения команды `unmount` вы можете снова деактивировать криптоустройство (`luksClose`), чтобы открыть доступ к `/dev/sdh1`. Только после этого можно извлечь флешку.

```
root# cryptsetup luksFormat /dev/sdh1
Безвозвратно заменить данные /dev/sdh1
Вы уверены? (Введите YES): YES
Введите пароль LUKS: **********
Подтвердите пароль: **********
Команда выполнена успешно.
root# cryptsetup luksOpen /dev/sdh1 mycontainer
Введите пароль LUKS: **********
root# mkfs.ext3 /dev/mapper/mycontainer
root# mount /dev/mapper/mycontainer /test
root# touch /test/xy
root# umount /test/
root# cryptsetup luksClose mycontainer
```

Разумеется, вместо USB-флешки можно использовать и раздел внешнего или внутреннего жесткого диска, устройство RAID или логический том вашей системы LVM. Для этого просто замените `/dev/sdh1` названием устройства раздела или логического тома.

По умолчанию `cryptsetup` применяет алгоритм шифрования AES, длина ключа в котором составляет 128 бит. Вы можете в этом убедиться, воспользовавшись командой `cryptsetup luksDump`, которая выдает метаинформацию о криптоустройстве; LUKS сохраняет эту метаинформацию в специальном секторе носителя данных.

```
root# cryptsetup luksDump /dev/sdh1
LUKS header information for /dev/sdh1
Version:        1
Cipher name:    aes
Cipher mode:    cbc-essiv:sha256
Hash spec:      sha1
```

```
Payload offset: 1032
MK bits:        128
...
```

Если хотите использовать другой алгоритм шифрования или более длинный ключ, сообщите нужные данные с помощью параметров `-c` и `-s` команде `cryptsetup luksFormat`. Чтобы узнать, какие алгоритмы можно применить, посмотрите `cat /proc/crypto`. В настоящее время наиболее надежными считаются алгоритмы AES и TwoFish. Обратите внимание, что алгоритмы шифрования сейчас активно исследуются и ситуация с ними быстро меняется: они часто оказываются не такими надежными, как это казалось поначалу.

Используя команду `cryptsetup luksAddKey`, можно обезопасить доступ к устройству LUKS с помощью восьми различных паролей. Это позволяет совместно работать с носителем, для доступа к которому каждый из пользователей применяет собственный пароль.

Команда luksformat. Эта команда немного упрощает создание зашифрованного раздела или носителя данных. Сначала она выполняет `cryptsetup luksFormat`, а затем `mkfs.vfat`. Если вы хотите работать с файловой системой другого типа, укажите его с помощью параметра `-t`.

СОВЕТ

После выполнения команды (зачастую и при возникновении ошибки) криптоустройство /dev/mapper/luksformatn может не сразу отреагировать. Прежде чем удалить носитель данных и еще раз попытаться создать криптоустройство, выполните команду cryptsetup luksClose luksformat*n*.

Работа с локальным компьютером. Если вы подключаете к компьютеру внешний носитель данных, отформатированный под LUKS, и начинаете работать с Gnome или KDE, то носитель данных автоматически распознается как криптоустройство. Открывается окно, в котором вам следует указать пароль для шифрования (рис. 14.3). После этого носитель данных подключается к файловой системе. Имя контейнера для `/dev/mapper` — `luks_crypto_`*`uuid`*. При отключении носителя нужно выполнить `luksClose`.

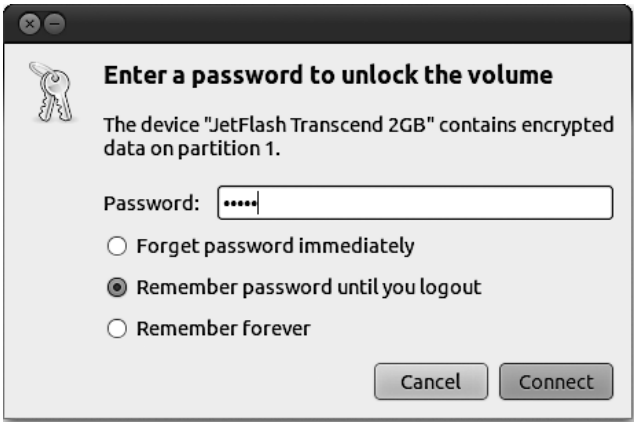


Рис. 14.3. Использование криптоустройства в Gnome

Файл crypttab. Если вы создали зашифрованную файловую систему в разделе локального диска, то, вероятно, захотите, чтобы эта система подключалась к дереву каталогов при запуске компьютера. В пакете `cryptsetup` уже содержатся сценарии, необходимые для автоматизации такого процесса. Для работы сценариев необходимо, чтобы криптоустройство было указано в файле `/etc/crypttab`.

Файл построен очень просто: в первом столбце указывается название для `/dev/mapper`, во втором — имя устройства, в третьем — файл, из которого должен быть считан ключ (например, с USB-флешки), или `none`, если пароль шифрования вводится в интерактивном режиме; в четвертом же столбце приводятся параметры.

В приведенном ниже примере нужно создать устройство `/dev/sda7`, которое имеет имя `/dev/mapper/cdisk1`. Пароль необходимо указать при запуске компьютера, а криптоустройство создается с LUKS (остальные параметры описаны в `man crypttab`).

```
# Файл /etc/crypttab
# Название модуля  Устройство  Файл ключей  Параметры
cdisk1             /dev/sda7   none         luks
```

Для того чтобы не только активизировать криптоустройство, но и подключить файловую систему к дереву каталогов, необходимо дополнить `/etc/fstab`. Следующая строка позволяет использовать файловую систему через каталог `/media/private-data`:

```
# Файл /etc/fstab
...
/dev/mapper/cdisk1   /media/private-data   ext3   defaults   0 0
```

После этого перезапустите компьютер и проверьте, все ли работает так, как нужно.

Недостатки. Шифрование раздела имеет также недостатки. Во-первых, все операции с файлами совершаются заметно медленнее, чем обычно — и более того, тем медленнее, чем сложнее (и надежнее) применяемый метод шифрования. По возможности используйте быстрый процессор с несколькими ядрами. Во-вторых, при загрузке системы необходимо вводить пароль. Это не просто неудобно — главное, что в ваше отсутствие компьютер нельзя будет перезапустить. В принципе с зашифрованными разделами стоит работать только на локальном компьютере и в исключительных случаях на сервере.

TrueCrypt. У представленных здесь методов, основанных на dm_crypt и LUKS, есть интересная альтернатива — TrueCrypt (http://www.truecrypt.org/). Эта шифровальная программа доступна также для Windows и Mac OS X, таким образом, она значительно упрощает обмен данными между Linux и другими системами. Исходный код открыт, но некоторые его части несовместимы с лицензией GPL, поэтому в распространенных дистрибутивах TrueCrypt отсутствует.

Ecryptfs (Ubuntu). В Ubuntu есть возможность зашифровать весь домашний каталог. Зашифрованный каталог автоматически подключается к файловой системе, а при выходе снова отключается. Внутри системы для этого применяется не dm_crypt и LUKS, а файловая система ecryptfs.

Шифрование целой системы

Файл /etc/crypttab, рассмотренный выше, позволяет сделать еще один небольшой шаг от шифрования раздела локального диска (например, /home) к шифрованию всей системы, в том числе системного раздела. Важно учесть две детали: во-первых, GRUB не может получить доступ к зашифрованным данным, поэтому обязательно должен быть отдельный, незашифрованный загрузочный раздел. Во-вторых, для доступа к системному разделу нужно ввести пароль шифрования; необходимые для этого функции должны быть интегрированы в виде сценариев в файл Initrd (все необходимые файлы содержатся в пакете cryptsetup).

Отдельно необходимо рассказать, кому вообще может потребоваться зашифровать всю систему: в принципе будет вполне достаточно зашифровать только личные файлы, находящиеся в каталоге /home. Однако для вора, укравшего ноутбук и желающего заполучить содержащиеся в нем данные, может быть очень информативен и системный раздел: в /var/cache или /var/tmp могут содержаться остатки отосланных электронных писем, распечатанных документов, прочитанных файлов PDF и т. д.; в /var/log документируется, кто и когда работал с компьютером; в разделе подкачки содержатся всевозможные выгруженные блоки данных с информацией, важной с точки зрения безопасности, и т. д. Таким образом, если вы хотите максимально защитить данные (например, личные), хранящиеся на компьютере, попробуйте зашифровать всю систему.

Установка. В большинстве крупных дистрибутивов в установочной программе предусмотрена функция, позволяющая зашифровать всю файловую систему. В любом случае в дистрибутивах, где на выбор предлагается несколько способов или видов установки, нужно, как правило, останавливаться на традиционном варианте; установочные программы, запускаемые с «живого» CD или DVD, в силу принципа их работы не подходят. В следующем списке перечислены подходящие средства установки для важнейших дистрибутивов:

- Debian — все стандартные CD или DVD (в том числе Netinstall, но не «живые диски»);
- Fedora — установка с DVD (но не «живые диски»);
- openSUSE — инсталляция с DVD в версии 11.2 и выше (но не «живые диски»);
- Ubuntu — установка с CD Alternate в текстовом режиме (но не с CD для локальной системы).

В Debian и Ubuntu выберите вариант секционирования диска Шифрование ▸ LVM-система, в Fedora откройте диалоговое окно секционирования, а в нем установите флажок Зашифровать систему. В openSUSE при секционировании укажите параметр На базе менеджера логических томов и С шифрованием.

Организация системы. В большинстве дистрибутивов зашифрованные системы построены одинаково: для GRUB создается незашифрованный загрузочный раздел, а другой раздел — зашифрованный — работает как физический том для LVM. Таким образом, все разделы, созданные с применением LVM (раздел подкачки, системный раздел, разделы с данными) автоматически шифруются. Кроме того, не требуется задавать для каждого раздела собственный пароль; достаточно будет общего паро-

ля для всей системы LVM. На рис. 14.4 схематически показано строение такой системы, причем обозначения устройств и логических томов взяты из Fedora.

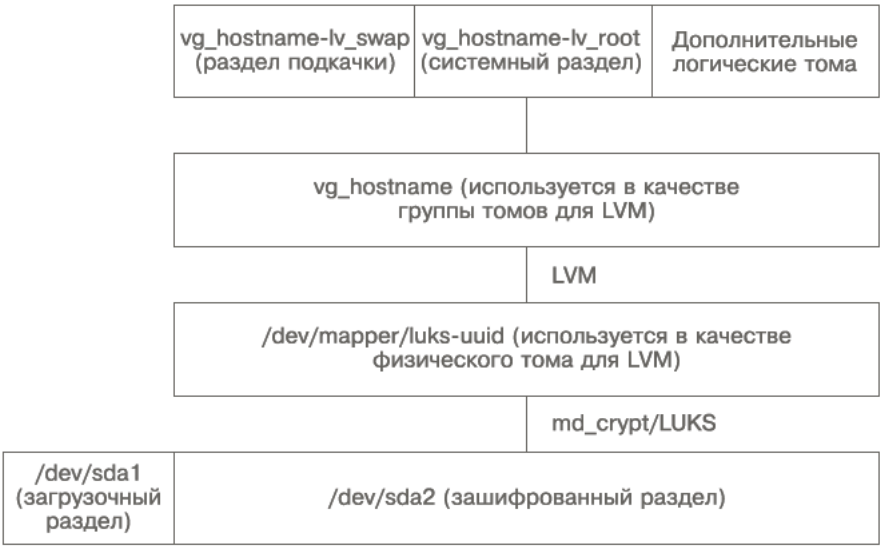


Рис. 14.4. Полностью зашифрованная система Linux

Конечно, вариант организации системы, показанный на рис. 14.4, далеко не единственный. Еще один вариант позволяет отказаться от LVM и отдельно зашифровать каждый раздел. При этом для раздела подкачки можно использовать случайный ключ, который будет заново создаваться в `/dev/urandom` при каждом запуске системы.

Можно представить себе и другой метод задания ключа: не вводить пароль интерактивно, а считывать ключ из файла, расположенного на USB-флешке, при загрузке системы. В таком случае флешка служит своего рода аппаратным ключом, необходимым для загрузки компьютера. Некоторые устройства для чтения карт памяти также могут работать с Linux; однако интеграция подобных устройств с системой шифрования выполняется вручную.

Шифрование твердотельных дисков. Вообще, все описанные выше методы шифрования в равной мере применимы при работе как с жесткими, так и с твердотельными дисками. Однако учитывая конструктивные особенности твердотельного диска, шифрование файловой системы на нем связано с двумя серьезными недостатками.

- Из соображений безопасности приходится всегда шифровать все содержимое файловой системы, то есть и те блоки данных, которые в настоящее время не используются. И, опять же, с точки зрения безопасности, нецелесообразно уведомлять твердотельный накопитель о наличии свободных блоков с помощью TRIM. В долгосрочной перспективе производительность SSD при таком шифровании будет снижаться.

- Многие SSD пытаются предварительно сжимать сохраняемые данные. Это особенно касается твердотельных дисков с широко распространенным контроллером Sandforce. Но сжимать зашифрованные данные невозможно, и у подобных SSD при работе с зашифрованной информацией будет существенно снижаться скорость работы.

Наилучшее решение этой дилеммы заключается в использовании таких твердотельных дисков, которые позволяют осуществлять шифрование на аппаратном уровне. Контроль шифрования выполняет не сама операционная система Linux, а EFI или BIOS материнской платы. Тем не менее, весьма сложно судить, насколько надежны алгоритмы шифрования, встроенные в сам SSD.

15 GRUB

Программа GRUB — это так называемый загрузчик. Она запускается первой после включения компьютера и, как правило, отображает меню, в котором можно выбрать для загрузки операционную систему Windows или какой-либо дистрибутив Linux. В большинстве современных дистрибутивов применяется версия GRUB 2. В старых, особенно корпоративных, установках Linux, вы наверняка столкнетесь с устаревшей, не поддерживаемой уже несколько лет версией GRUB 0.97. Поэтому я кратко расскажу и об этой версии.

15.1. Основы

Аббревиатура GRUB означает Grand Unified Bootloader (Грандиозный унифицированный загрузчик). Программа запускает Linux и вообще очень полезна для работы с параллельными установками Windows и Linux, так как непосредственно после запуска компьютера позволяет выбрать для работы желаемую операционную систему.

Версии GRUB. С июня 2012 года используется версия GRUB 2.0. Разработка этой версии велась еще с 2005 года. Некоторые дистрибутивы не стали дожидаться полной доработки GRUB 2 и уже давно применяют новый загрузчик. Например, Ubuntu использует GRUB 2, начиная с версии 9.10, а Debian — с версии 6. Дистрибутив openSUSE перешел на GRUB 2 только в версии 12.2. Особый случай представляет дистрибутив Fedora: его разработчики перешли на GRUB 2 уже в 16 версии, но только в компьютерах, работающих с BIOS. Если же инсталлировать Fedora 16 или 17 с EFI, то вместе с дистрибутивом будет установлена EFI-совместимая версия GRUB 0.97!

Итак, версия, применявшаяся до GRUB 2, имеет название GRUB 0.97 *legacy* (устаревшая). Примечание «устаревшая» указывает на то, что эта версия не развивается уже несколько лет и ее технической поддержкой никто не занимается. А число 0,97 не оставляет сомнений, что данная разработка не достигла и никогда не достигнет уровня GRUB 1.0. Несмотря на это, производители многих дистрибутивов неоднократно применяли GRUB 0.97 legacy по собственному усмотрению, в результате реализация некоторых функций GRUB отличается от дистрибутива к дистрибутиву!

Хотя в новых дистрибутивах применяется преимущественно GRUB 2, GRUB 0.97 по-прежнему широко распространен. Во-первых, как уже было сказано, эта программа встретится вам в старых версиях Linux, во-вторых, в корпоративных дистрибутивах GRUB 0.97 сохраняет свои позиции основного загрузчика (например, в RHEL он задействуется вплоть до версии 6.3).

Что нового в GRUB 2. Далее перечислены все важнейшие нововведения, появившиеся в GRUB 2 по сравнению с официальной версией GRUB 0.97. В некоторых дистрибутивах имеется пропатченная версия GRUB 0.97, в которой присутствуют не все, а лишь некоторые из перечисленных ниже функций. Правда, такие пропатченные версии являются неофициальными и содержат особенности, специфичные для конкретных дистрибутивов.

- GRUB 2 совместим с менеджером логических томов и программным RAID. Таким образом, в установках LVM и RAID больше не требуется загрузочный раздел `/boot`.
- GRUB 2 совместим с файловыми системами ext4 и btrfs.
- GRUB 2 позволяет запрашивать разделы диска по уникальному идентификатору файловой системы.
- GRUB 2 совместим с системами, альтернативными BIOS, — EFI и coreboot (ранее называлась LinuxBIOS).
- GRUB 2 позволяет отображать в записях меню любые символы Unicode.
- Внутрисистемная реализация GRUB 2 выполнена совершенно по-новому. Дополнительные функции теперь реализованы в виде модулей, которые GRUB загружает на время использования. Такая организация должна обеспечить сравнительно простую расширяемость и техническую поддержку системы.
- Конфигурация GRUB стала значительно сложнее. Конфигурационный файл `grub.cfg` является результатом исполнения различных конфигурационных сценариев. `grub.cfg` может и сам содержать сценарный код на shell-подобном языке либо на языке программирования LUA.

Компоненты и пакеты. Файлы, необходимые для работы GRUB 2, как правило, распределены по нескольким пакетам. Так, в Debian и Ubuntu в пакете `grub-common` содержатся различные платформозависимые конфигурационные файлы и команды, а в пакете `grub-pc` находятся файлы, специфичные для BIOS. Для компьютеров, использующих вместо BIOS EFI, coreboot и другие программы, вместо `grub-pc` требуются, соответственно, пакеты `grub-efi-amd64`, `grub-efi-ia32` или `grub-coreboot`. Наконец, в пакете `grub-rescue-pc` имеется IMG-файл и файл образа диска, необходимые для того, чтобы сохранить систему аварийного восстановления GRUB на флешке или CD-диске. Таким образом, в критическом случае вы сможете запустить GRUB с такой флешки или диска, а потом запустить систему, вручную вводя команды GRUB или изменяя предусмотренные записи меню.

В Fedora удалось избежать такого разброса программ по разным пакетам. В пакете `grub2` содержится GRUB для компьютеров, работающих с BIOS, а в пакете `grub2-efi` — версия, совместимая с EFI. Обновлением конфигурации GRUB (этот процесс необходимо выполнять после появления обновлений для ядра) занимаются Fedota-специфичные сценарии из пакета `grubby`.

Независимо от конкретного дистрибутива, для работы GRUB 2 требуется предварительно установить пакет `os-prober`. Одноименная команда просматривает все доступные разделы дисков на наличие операционных систем. Результат выполнения `os-prober` интегрируется в автоматически создаваемое меню GRUB.

Отличия, специфичные для конкретных дистрибутивов. К сожалению, в различных дистрибутивах GRUB 2 не только распределяется по разным пакетам, но и работает на основе не совсем идентичных конфигурационных файлов и механизмов. В этой главе описываются Debian версии 6 и выше, Fedora версии 16 и выше, openSUSE версии 12.2 и выше и Ubuntu версии 9.10 и выше.

Документация. Официальная документация по GRUB выдается по запросу справочной командой `info grub`. Более подробная информация содержится на официальной странице GRUB: http://www.gnu.org/software/grub/.

Более подробная информация по GRUB приводится на следующих сайтах:

- http://www.dedoimedo.com/computers/grub-2.html — неофициальный учебник;
- http://lists.gnu.org/archive/html/grub-devel/ — почтовый архив по GRUB;
- http://fedoraproject.org/wiki/Features/Grub2 — Fedora-специфичные детали;
- https://help.ubuntu.com/community/Grub2 — Ubuntu-специфичные детали.

Загрузка системы в BIOS

Прежде чем подробно обсудить установку и конфигурацию GRUB, коротко рассмотрим, что же происходит при загрузке системы. В зависимости от того, работает ли на компьютере традиционная система BIOS или более новая EFI, процесс загрузки может значительно отличаться.

После включения компьютера с BIOS инициализируется *базовая система ввода-вывода*. В ходе этого процесса на экране обычно отображается несколько системных сообщений — например, каким объемом памяти располагает компьютер. Затем BIOS загружает содержимое первого сектора первого жесткого диска в память и выполняет код этого сектора. Такой специальный сектор называется MBR (Master Boot Record, главная загрузочная запись).

ВНИМАНИЕ

На вашем жестком диске может быть только одна главная загрузочная запись, но вполне может присутствовать несколько операционных систем. Разумеется, здесь есть потенциал для конфликтов! MBR перезаписывается как при установке Linux, так и при установке Windows. GRUB вполне может запускать и Windows, но Windows не учитывает возможного наличия Linux на диске. Поэтому после установки Linux GRUB приходится «ремонтировать». Для этого лучше всего воспользоваться «живым диском» или системой аварийного восстановления. Поэтому сначала лучше устанавливать Windows, и уже потом — Linux. Если позже вы снова установите Windows и перепишете таким образом главную учетную запись, то советы, как справиться с данной проблемой, приведены в пункте «Ремонт GRUB с помощью "живого диска" (на компьютерах с BIOS)» подраздела «Установка вручную и первая помощь при работе с компьютерами с BIOS» раздела 15.3 (для GRUB 2) и в подразделе «Исправление установки GRUB с помощью "живого диска"» раздела 15.4 (GRUB 0.97).

Загрузчик Windows. Если на компьютере установлена ОС Windows, то в MBR располагается крошечная программа. Она находит раздел, помеченный как «активный», а затем запускает загрузчик Windows, который расположен в загрузочном

секторе данного раздела. Если на компьютере установлено несколько версий Windows, то загрузчик Windows позволяет выбрать одну из них.

Загрузчик Linux. Если на компьютере установлена и Linux, MBR обычно заменяется кодом загрузчика GRUB. В таком случае GRUB может либо запустить Linux, либо совершить условный переход для запуска Windows.

Альтернативный метод заключается в том, чтобы не трогать MBR и установить GRUB в загрузочном секторе системного раздела Linux, пометив этот раздел как «активный». Такой метод хотя и не противоречит правилам MBR, менее надежен и поэтому почти не используется.

Основная загрузочная запись имеет размер всего 512 Мбайт, поэтому загрузчик не сможет полностью поместиться в MBR. Чтобы можно было обойти это ограничение, MBR может вместить ровно такой фрагмент кода, который позволит вам запустить оставшуюся часть загрузчика уже с жесткого диска. Соответственно, код GRUB подразделяется на две или три части: `stage1` находится в основной загрузочной записи и предназначена для того, чтобы загрузить первые сектора части `stage1_5` или `stage2`. В части `stage1_5` содержится дополнительный код, обеспечивающий доступ к файлам различных файловых систем. Наконец, в `stage2` содержится сам загрузчик.

Когда запустится загрузчик, появится меню, в котором будут на выбор представлены все операционные системы, определенные в процессе конфигурации GRUB (обычно это Windows и Linux). Теперь с помощью клавиш управления курсором можно выбрать интересующую вас операционную систему и запустить ее, нажав Enter. Часто GRUB настроен так, чтобы по истечении определенного промежутка времени операционная система загружалась автоматически.

Запуск Linux. Если вы определяете в загрузчике, что необходимо запустить Linux, то загрузчик должен поместить в оперативную память файл ядра Linux и запустить этот файл. Обычно файл ядра Linux называется `/boot/vmlinuz` (последняя буква `z` указывает на то, что ядро архивировано). Иными словами, загрузчик должен быть в состоянии загрузить из файловой системы весь указанный файл.

Параметры ядра. Обычно ядру сообщается несколько параметров. Как минимум один параметр передается обязательно: имя устройства системного раздела (например, `root=/dev/sdb13`). Это делается для того, чтобы ядро знало, какой из разделов является системным. Когда ядро запустится, управление будет передано программе Linux `/sbin/init`, отвечающей за инициализацию системы Linux и подробно описанной в разделе 16.1. Например, она отвечает за запуск сетевых служб.

Доступ к модулям ядра. Ядро Linux имеет модульную структуру. Это означает, что в самом ядре содержатся только самые элементарные функции. Дополнительные функции, необходимые для доступа к компонентам аппаратного обеспечения, для считывания и изменения файлов из различных файловых систем и т. д., напротив, находятся в модулях, которые при необходимости загружаются из файловой системы и, таким образом, дополняют ядро.

Чтобы процесс запуска прошел успешно, ядро должно иметь возможность получать доступ к системному разделу. Если этот раздел расположен в файловой системе, не поддерживаемой ядром напрямую, или на жестком диске SCSI, для которого в ядре нет подходящего драйвера, возникает проблема «курицы и яйца»:

ядро не может обратиться к файловой системе и загрузить из нее модули, которые понадобились бы ядру, чтобы прочитать файлы файловой системы.

Файл Initrd. Решение заключается в том, что GRUB должен загружать не только ядро, но и файл `Initrd`. Это специальный файл, в котором содержатся все модули ядра, необходимые для запуска системы. Ядро временно использует этот файл как псевдодиск, то есть оно может загрузить все модули сразу же после запуска псевдодиска (сокращение Initrd означает Initial RAM Disk — диск в оперативной памяти для начальной инициализации).

Обычно файл `Initrd` называется `/boot/initrd` или `/boot/initrd.gz`. В большинстве дистрибутивов имеются инструменты, позволяющие создать такой файл `Initrd`, который бы подошел к применяемому аппаратному обеспечению и к файловой системе, которая находится в системном разделе (команда `mkinitrd`).

Установка и конфигурация GRUB. Когда на страницах этой книги говорится об установке программы, обычно имеется в виду установка программного пакета на жесткий диск. В этой главе все иначе. Под установкой GRUB понимается процесс, в ходе которого стартовый код GRUB записывается в загрузочный сектор жесткого диска.

Конфигурация GRUB протекает по-разному в зависимости от версии GRUB. В GRUB 0.97 применяется центральный конфигурационный файл `/boot/grub/menu.lst`. В GRUB 2 есть целая коллекция конфигурационных сценариев, находящихся в каталоге `/etc/grub.d/`. Благодаря исполнению этих сценариев создается сам конфигурационный файл GRUB 2 `/boot/grub/grub.cfg`.

Запуск системы с EFI

В Apple система EFI (Extensible Firmware, расширяемый интерфейс между операционной системой и микропрограммами) применяется уже много лет, но на рынке ПК с другими системами постепенный отказ от BIOS продлился значительно дольше. Только с выходом на рынок Windows 8 можно ожидать, что EFI будет функционировать на каждом ПК. (Разумеется, Windows 8 можно устанавливать и на более старые компьютеры, работающие с BIOS.)

EFI и GPT. Даже притом, что EFI заново разрабатывался практически с нуля, выгоды для конечного пользователя остаются довольно ограниченными. Но не может не радовать, что теперь есть гигантские жесткие диски с таблицей секционирования GUID (GPT), совершенно не доставляющие проблем при работе. Не забывайте, что применение GPT не является обязательным, в том числе и на компьютерах с EFI! EFI нормально работает и с такими жесткими и твердотельными дисками, чья таблица секционирования сохранена в главной загрузочной записи (MBR). Лично я считаю, что использовать GPT предпочтительно и на жестких и твердотельных дисках, которые не превышают по размеру 2 Тбайт, поскольку такая таблица значительно упрощает весь процесс секционирования диска.

Параллельная установка. С введением EFI значительно изменился весь загрузочный процесс. В то время как BIOS в принципе предусматривал установку *только одной* операционной системы и при любой параллельной установке приходилось задействовать менеджер загрузок, EFI сам по себе поддерживает установку

нескольких операционных систем. Каждая операционная система может сохранить собственный загрузчик в специально предусмотренном для этого сегменте диска. При запуске компьютера сначала происходит инициализация EFI, а потом автоматически запускается загрузчик той операционной системы, которая была задана по умолчанию. (Как правило, по умолчанию задается та операционная система, которая была установлена последней.) Если при запуске компьютера нажать специальное сочетание клавиш (к сожалению, оно не стандартизировано и варьируется в зависимости от производителя EFI и ПК), то EFI отобразит меню со всеми загрузчиками.

Таким образом, при работе с EFI GRUB требуется лишь для того, чтобы запустить Linux. Вторая функция GRUB — выбор одной из нескольких операционных систем — с EFI становится уже излишней (хотя меню GRUB — исключительно полезная вещь даже при работе с EFI).

В сущности, GRUB с EFI совершенно не нужен. Ядро Linux содержит все необходимые функции, позволяющие EFI-системе запускать Linux напрямую, то есть, *минуя* GRUB или другие загрузчики. Но на практике этот вариант пока не играет роли: GRUB или другие загрузчики позволяют выбирать одну из нескольких установленных версий ядра, передавать параметры ядра, изменять параметры загрузки и т. д. При запуске Linux непосредственно из EFI вся эта гибкость теряется.

EFI и GRUB. Чтобы EFI и GRUB правильно взаимодействовали, нужно настроить специальную версию EFI, рассчитанную на работу с GRUB. Установочные программы распространенных дистрибутивов решают эту задачу автоматически. В любом случае такое взаимодействие предполагает, что установочная программа должна выполняться в EFI-режиме! (Многие варианты реализации EFI совместимы с BIOS. При запуске той или иной установки Linux решающее значение имеет то, в каком режиме происходит данная установка. На некоторых компьютерах один установочный носитель — то есть DVD или флешка — в меню EFI отображается как два: одно устройство в BIOS-режиме и одно в EFI-режиме. Вы должны выбирать именно ту запись, которая относится к EFI!)

Раздел EFI. На компьютерах с EFI код GRUB записывается не в главную загрузочную запись, а в каталог раздела с EFI. Это особый раздел диска, в котором применяется файловая система VFAT. Такой раздел должен быть помечен специальным идентификатором: (MBR) или (C12A7328-F81F-11D2-BA4B-00A0C93EC93B) (GPT).

Microsoft рекомендует создавать EFI-раздел как первый раздел жесткого диска, хотя стандарт EFI этого и не требует. Данный раздел не должен быть очень большим, вполне достаточно будет 100–200 Мбайт (в тех дистрибутивах, которые я протестировал, загрузчик EFI в каждом из случаев требовал менее 1 Мбайт. Windows 8 требует значительно больше места — ориентировочно порядка 25 Мбайт).

Раздел с EFI должен быть подключен к файловой системе Linux в каталоге `/boot/efi`! Если при установке Linux вы выполняете секционирование диска вручную, то об этом подключении должны позаботиться вы сами.

Ядро, файл Initrd. Если GRUB уже работает, то запуск Linux протекает точно так же, как и на компьютере с BIOS: GRUB загружает ядро и запускает его, и в ходе запуска происходит передача файла Initrd и параметров ядра.

Файлы Initrd

Ядро Linux содержит разнообразные модули. Многие дополнительные функции — такие как управление SCSI-картой, доступ к определенным файловым системам, массивам RAID или сегментам LVM — обеспечиваются не самим ядром, а его модулями. Но при запуске системы это доставляет определенные проблемы. Действительно, как ядру загрузить модуль, если оно даже не имеет возможности получить доступ к файловой системе. Поэтому модули, которые требуются непосредственно для запуска системы, помещаются в диск оперативной памяти для начальной инициализации (Initial RAM Disk, сокращенно Initrd). Соответствующий Initrd-файл передает GRUB ядру (обратите внимание на ключевое слово `initrd` в конфигурационном файле GRUB).

Ядро и файл Initrd обычно находятся в каталоге `/boot`, но название самого Initrd-файла различается в зависимости от дистрибутива. В принципе, если применяется GRUB 0.97, то каталог `/boot` часто находится в собственном сегменте диска, вне системы RAID или LVM, поскольку GRUB 0.97 может обращаться только к обычным сегментам жесткого диска. В этом отношении GRUB 2 отличается значительно большей гибкостью.

Файл Initrd должен содержать такие модули ядра, версия которых точно совпадает с версией ядра. Поэтому всякий раз, когда вы устанавливаете или самостоятельно компилируете новую версию ядра, необходимо заново создавать соответствующий этому ядру Initrd-файл. При обновлении ядра этот процесс обычно автоматически выполняет сама программа обновления. А если вы самостоятельно устанавливаете ядро, то сами должны позаботиться и об Initrd-файле.

Строго говоря, название «Initrd-файл» в большинстве современных дистрибутивов уже неприменимо. На самом деле мы работаем с `initramfs`-файлами, организация которых будет описана немного ниже. Но поскольку и в параметрах GRUB, и в различных командах для создания файлов используется понятие `initrd`, а само ядро правильно интерпретирует файл, несмотря на неверное название, я тоже не буду отступать в книге от привычного названия Initrd — пусть оно и заведомо неверное.

Файл Initrd не всегда необходим для работы. Если ядро вашего дистрибутива содержит все компоненты, необходимые для загрузочного процесса, то запуск возможен и без Initrd-файла. Но для этого ядро нужно соответствующим образом скомпилировать — и именно такого в большинстве дистрибутивов не происходит. (Почти все модули ядра при желании можно интегрировать в само ядро. Разумеется, файл ядра от этого увеличится.)

К огромному сожалению, создание Initrd-файлов не стандартизировано. В каждом дистрибутиве для этого используются собственные инструменты. Initrd-файлы содержат не только модули ядра, но и сценарии для инициализации оборудования. Исполнение этих сценариев длится достаточно долго и стало головной болью для многих разработчиков, так как сильно тормозит процесс загрузки.

Update-initramfs (Debian, Ubuntu). В Debian и Ubuntu для создания Initrd-файлов и их последующего администрирования предусмотрен сценарий `update-initramfs`. В простейшем случае для обновления Initrd-файла новейшей установленной версии ядра вы просто задаете параметр `-u`. Но если вы хотите обновить

Initrd-файл не до новейшей, а до какой-то другой версии ядра, то укажите номер версии с помощью параметра -k. Параметр -k all обновляет Initrd-файлы для всех установленных версий ядра.

С помощью параметров -c или -d сценарий update-initramfs создает новый или удаляет имеющийся Initrd-файл. В таком случае обязательно требуется указать версию ядра с помощью параметра -k.

```
root# update-initramfs -c -k 3.5-13-generic
update-initramfs: Generating /boot/initrd.img-3.5-13-generic
```

«За кулисами» сценарий update-initramfs обращается для создания Initrd-файлов к сценарию mkinitramfs. Базовая конфигурация происходит в файле /etc/initramfs-tools/initramfs.conf, Кроме того, к Initrd-файлу добавляются все модули, перечисленные в каталоге /etc/initramfs-tools/modules (один модуль на строку).

В стандартной конфигурации (с MODULES=most в initramfs.conf) сценарий mkinitramfs создает довольно крупные Initrd-файлы, содержащие многочисленные дополнительные модули. В частности, здесь находятся важнейшие файловые системы Linux, драйверы для USB, SCSI и SATA, а также сетевые и NFS-драйверы.

При непосредственном вызове mkinitramfs (это приходится делать редко), нужно как минимум сообщить имя нового Initrd-файла (параметр -o). Если Initrd-файл должен быть создан не для актуальной версии ядра, то дополнительно укажите желаемую версию:

```
root# mkinitramfs -o myinitrd 3.5-13-generic
```

Dracut (Fedora, Red Hat). Fedora в версии 12 и выше и Red Hat Enterprise Linux в версии 6 и выше используют для создания Initrd-файла программу dracut. В более ранних версиях Fedora и RHEL вместо этого применяется программа mkinitrd.

dracut автоматически выполняется при каждом обновлении ядра. Initrd-файлы, создаваемые с помощью dracut, содержат меньше дистрибутиво-специфичного кода, чем ранее, и в большей мере опираются на информацию из системы udev. Таким образом, техническая поддержка инфраструктуры dracut должна быть проще, чем ранее, а сама система должна запускаться быстрее. Но эти аргументы пока не убедили производителей других дистрибутивов в целесообразности перехода на dracut.

Команда dracut учитывает настройки, указанные в файле /etc/dracut.conf, и встраивает в Initrd-файл модули из каталога /usr/share/dracut/modules.d. Чтобы создать Initrd-файл для самостоятельно скомпилированного ядра версии 3.5.3 (файл /boot/vmlinuz-3.5.3), выполните следующую команду:

```
root# dracut /boot/initrd-3.5.3    3.5.3
```

Подробнее о dracut рассказано на сайте https://dracut.wiki.kernel.org/index.php/Main_Page.

Mkinitrd (SUSE). В дистрибутивах SUSE Initrd-файлы создаются с помощью команды mkinitrd. Как правило, этой команде не требуется передавать никаких параметров. Команда mkinitrd автоматически создает Initrd-файлы для всех файлов ядра, которые находит в каталоге /boot. Новые Initrd-файлы получают имя вида /boot/initrd-nnn, где nnn — версия ядра. Кроме того, mkinitrd создает ссылку, которая указывает с /boot/initrd на Initrd-файл, соответствующий vmlinuz.

Если вы хотите создать только один определенный Initrd-файл, то с помощью параметров `-k` и `-i` можете задать, соответственно, файл ядра и файл Initrd (по умолчанию, это делается в каталоге `/boot`). Команда `mkinitrd` интерпретирует переменную `INITRD_MODULES` из файла `/etc/sysconfig/kernel`. В этом файле содержатся все модули, необходимые для загрузки, он может иметь примерно следующий вид:

```
# в /etc/sysconfig/kernel
INITRD_MODULES="thermal ahci ata_piix ata_generic processor fan"
```

Дополнительные модули указываются с помощью `-m`. Более подробная информация по `mkinitrd` выводится с помощью параметра `-h`, справки `man mkinitrd`, а также в исходном коде самого сценария (файл `/sbin/mkinitrd`).

Обновления ядра. Когда в дистрибутиве выполняется обновление ядра и при этом имя ядра меняется, необходимо соответствующим образом изменить и файл меню GRUB, а также создать Initrd-файл, подходящий для нового ядра. Во всех распространенных дистрибутивах эти операции выполняются автоматически в процессе обновления, и после перезапуска компьютера уже используется новое ядро. (В некоторых дистрибутивах в меню GRUB может также сохраняться и запись для старого ядра. Она пригодится, если с обновлениями возникнут проблемы, и систему придется далее использовать со старым ядром.)

Просмотр Initrd-файла. В версии ядра 2.6 и выше Initrd-файлы внутрисистемно представлены как файлы `initramfs`. Initrd-файл — это сжатый файл-архив (`cpio`), состоящий из различных каталогов и файлов. Если вы хотите просмотреть содержимое этого архива, сделайте так:

```
root# cd /boot
root# cp initrd-n.n initrd-test.gz
root# gunzip initrd-test
root# mkdir test
root# cd test
root# cpio -i < ../initrd-test
root# ls -lR
```

Ссылки. Большое количество информации о внутреннем строении системы `initramfs` приводится на сайтах http://www.kernel.org/doc/Documentation/filesystems/ramfs-rootfs-initramfs.txt и http://lwn.net/Articles/191004/.

15.2. Работа с GRUB (с точки зрения пользователя)

После установки Linux при перезапуске компьютера отображается меню для выбора нужной операционной системы (рис. 15.1). В зависимости от конфигурации внешний вид окна GRUB может быть довольно разным. В некоторых дистрибутивах GRUB имеет спартанский облик. Чтобы можно было использовать различные дополнительные функции загрузчика, покиньте графический режим, нажав клавишу Esc. В других дистрибутивах меню GRUB по возможности вообще не отображается. Если, например, на вашем компьютере установлена только одна операционная система — Ubuntu, то вы увидите меню GRUB, только если нажмете любую клавишу во время запуска компьютера.

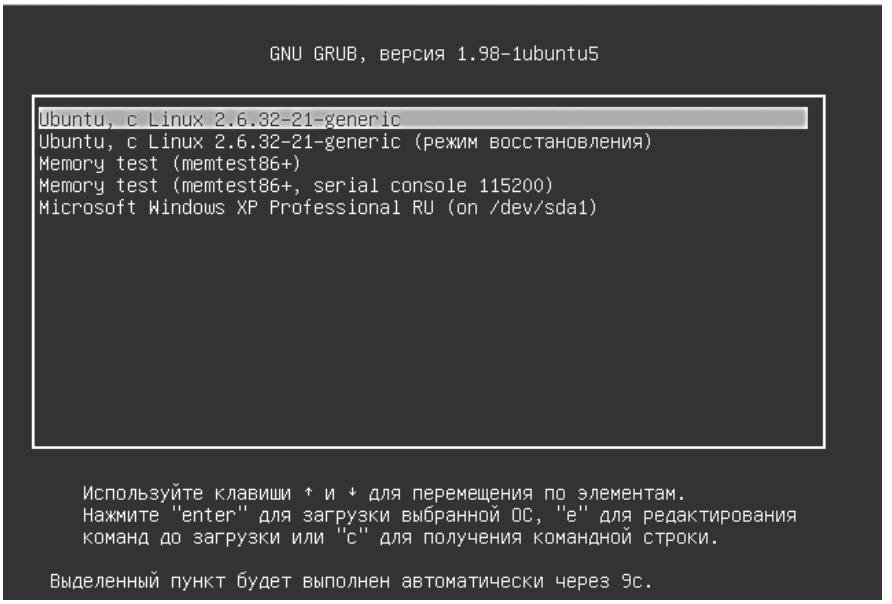


Рис. 15.1. Меню GRUB

Пароль. GRUB можно защитить паролем. В этом случае интерактивными функциями GRUB можно будет воспользоваться лишь тогда, когда будет нажата клавиша P и после этого будет введен пароль.

Раскладка клавиатуры. Как правило, в GRUB используется раскладка клавиатуры для США.

Передача параметров загрузки ядра Linux. Если меню GRUB не защищено паролем, то можно выбрать клавишами управления курсором определенную запись меню GRUB и изменить ее с помощью клавиши E (edit — «отредактировать»). После этого отобразится несколько строк, обладающих довольно своеобразным синтаксисом и описывающих, как должна быть запущена операционная система.

Функции редактирования используются в GRUB для того, чтобы при запуске Linux сообщить дополнительные параметры, касающиеся загрузки ядра (например, чтобы избежать аппаратных проблем). Нужно найти строку, которая будет выглядеть примерно так:

```
linux /boot/vmlinuz-n.n.n root=/dev/sdb13
```

В конце этой строки можно добавлять или изменять параметры. Чтобы потом запустить Linux с измененными параметрами, нажмите Ctrl+X или F10. Но такие изменения не сохраняются!

Интерактивное выполнение команд. Из меню GRUB можно перейти в режим интерактивного ввода команд — для этого нажмите C. В данном режиме пользователь вводит вручную различные команды. Таким образом, можно запустить операционную систему Linux даже тогда, когда нужная запись в меню GRUB отсутствует или является ошибочной. Чтобы интерактивно вводить команды, вы должны знать, в каком разделе диска находится Linux и как называются нужные вам команды

GRUB. Подробнее об этом — далее в этой главе. Но учитывайте, что синтаксисы GRUB 0.97 и GRUB 2 значительно различаются.

Например, следующие команды позволяют запустить дистрибутив Linux, установленный в разделе /dev/sdb13 с файловой системой ext4 (команда insmod ext2 загружает драйвер, поддерживающий файловые системы ext2, ext3 и ext4):

```
grub> insmod ext2
grub> set root='(hd1,13)'
grub> linux /boot/vmlinuz-n.n.n root=/dev/sdb13 ro
grub> initrd /boot/initrd-n.n.n
grub> boot
```

При вводе имен файлов GRUB с помощью клавиши Tab автоматически дополняются названия файлов из файловой системы, указанной администратором или находящейся на заданном диске. Введя команду cat, вы даже можете отобразить некоторые текстовые файлы. Кроме того, в командном режиме имеется еще несколько интересных возможностей работы с GRUB, но мы не будем их здесь рассматривать ради экономии места (команда help выводит список всех команд, а help commandname выдает информацию по конкретной команде).

Закрепление изменений, внесенных в меню. Как правило, GRUB считывает загрузочное меню из файла /boot/grub/menu.lst. GRUB 2, в свою очередь, использует для этого файл /boot/grub/grub.cfg. Данные файлы содержат команды, относящиеся к записям меню GRUB. Итак, если вы хотите закрепить изменения, внесенные в меню GRUB, вам нужно запустить Linux и изменить файл меню GRUB. В GRUB 0.97 можно обработать menu.lst прямо в редакторе. В GRUB 2 требуется изменять уже другие конфигурационные файлы и сгенерировать новую версию grub.cfg. Загрузчик автоматически примет сделанные изменения при перезапуске. Организация файла меню GRUB более подробно описана в следующих разделах этой главы.

15.3. GRUB 2

Базовая конфигурация

Grub.cfg. Меню GRUB определяется в файле grub.cfg, который в зависимости от дистрибутива может находиться в разных местах:

- /boot/grub/grub.cfg — файл меню GRUB в Debian и Ubuntu;
- /boot/grub2/grub.cfg — файл меню GRUB в Fedora и openSUSE версия 12.2 и выше;
- /etc/grub2.cfg — ссылка на /boot/grub2/grub.cfg в Fedora и openSUSE.

/etc/grub.d/. В принципе возможность вручную вносить изменения в файл grub.cfg не предусмотрена. Поэтому права доступа к этому файлу указаны как «только для чтения». Если вы хотите изменить меню GRUB, то измените лежащие в его основе конфигурационные файлы. Не может не радовать, что в Debian, Ubuntu, openSUSE и Fedora эти файлы находятся в одном и том же месте:

- /etc/grub.d/* — общие конфигурационные файлы GRUB;
- /etc/default/grub — дополнения, специфичные для конкретных дистрибутивов.

Update-grub. Чтобы заново сгенерировать эти файлы после внесения изменений или после обновления ядра, нужно выполнить одну из следующих команд:

```
root# update-grub                              (Debian/Ubuntu)
root# grub2-mkconfig -o /boot/grub2/grub.cfg(Fedora)
```

Пример grub.cfg. Результирующий файл grub.cfg выглядит наподобие приведенного ниже примера. Этот листинг автоматически создан в Ubuntu 12.10 с помощью grub.cfg и сокращен ради экономии места. Кстати, пусть вас не смущает строка insmod ext2: этот модуль GRUB отвечает за работу со всеми файловыми системами ext, в том числе с ext3 и ext4. Детали синтаксиса grub.cfg будут объяснены ниже в этом разделе:

```
# Пример /boot/grub/grub.cfg

# из /etc/grub.d/00_header
# Управление переменными
if [ -s $prefix/grubenv ]; then
   set have_grubenv=true
   load_env
fi
set default="0"
if [ "${prev_saved_entry}" ]; then
   set saved_entry="${prev_saved_entry}"
   save_env saved_entry
   set prev_saved_entry=
   save_env prev_saved_entry
   set boot_once=true
fi

# Определение различных функций
function savedefault { ... }
function recordfail { ... }
function load_video { ... }

# Выбор раздела с установочными файлами GRUB
insmod part_msdos
insmod ext2
set root='(hd0,msdos1)'
search --no-floppy --fs-uuid --set=root d45161e0-a9dd-421b-90e3-dc01887b140b
if loadfont /usr/share/grub/unicode.pf2 ; then
   set gfxmode=auto
   load_video
   insmod gfxterm
   insmod part_msdos
   insmod ext2
   set root='(hd0,msdos1)'
   search --no-floppy --fs-uuid --set=root d45161e0-a9dd-421b-90e3-
```

```
                                dc01887b140b
   set locale_dir=($root)/boot/grub/locale
   set lang=de_AT
insmod gettext
fi

terminal_output gfxterm
if [ "${recordfail}" = 1 ]; then
   set timeout=-1
else
   set timeout=10
fi

# из /etc/grub.d/10_linux
# Запуск Ubuntu
menuentry 'Ubuntu, mit Linux 3.5.0-2-generic' ... {
       recordfail
       gfxmode $linux_gfx_mode
       insmod gzio
       insmod part_msdos
       insmod ext2
       set root='(hd0,msdos1)'
       search --no-floppy --fs-uuid
                   --set=root 474ef599-7665-4fe0-a4fd-97df765db80e
       linux boot/vmlinuz-3.5.0-2-generic root=UUID=474e... ro quiet splash
       initrd /boot/initrd.img-3.5.0-2-generic
}

menuentry 'Ubuntu, с Linux 3.5.0-2-generic (режим восстановления)' ... {
       ...
       linux /boot/vmlinuz-3.5.0-2-generic root=UUID=474e...
                                ro recovery nomodeset
       ...
}

# из /etc/grub.d/20_memtest86+
# Тест памяти
menuentry "Memory test (memtest86+)" {
       insmod part_msdos
       insmod ext2
       set root='(hd0,msdos1)'
       search --no-floppy --fs-uuid --set=root d45161e0-a9dd-421b-90e3-
                                      dc01887b140b
       linux16 /boot/memtest86+.bin
}

# из /etc/grub.d/30_os-prober
# Запуск Windows
menuentry "Windows (loader) (on /dev/sda1)" {
       insmod ntfs
       set root='(hd0,1)'
```

```
        search --no-floppy --fs-uuid --set 2ca80f2ba80ef35e
        chainloader +1
}
```

Создание grub.cfg заново. В Debian и Ubuntu команда `update-grub` создает новую версию файла `grub.cfg`. В Fedora и openSUSE в аналогичном случае выполняется команда `grub2-mkconfig -o /boot/grub2/grub.cfg`. (Сценарий `update-grub`, специфичный для Debian/Ubuntu, так или иначе, содержит только эту команду.)

`grub[2]-mkconfig` интерпретирует описанные ниже конфигурационные файлы или сценарии. При этом, в частности в каталоге `/boot` меню GRUB, создаются записи для всех файлов ядра. Кроме того, просматриваются все доступные разделы жесткого диска. Если на диске есть и другие операционные системы, то записи в меню GRUB создаются и для них. Поэтому выполнение `update-grub` на компьютере с большим количеством сегментов на диске может длиться достаточно долго.

`grub[2]-mkconfig` автоматически выполняется при каждом обновлении ядра и удостоверяет, что в меню GRUB содержится новейшая версия ядра Linux.

/etc/default/grub. в файле `/etc/default/grub` содержатся некоторые глобальные настройки GRUB. Не забудьте, что вносимые здесь изменения вступают в силу лишь тогда, когда вы заново сгенерируете `grub.cfg`! В Ubuntu конфигурационный файл содержит следующие настройки:

```
# Файл /etc/default/grub
GRUB_DEFAULT=0
GRUB_HIDDEN_TIMEOUT=0
GRUB_HIDDEN_TIMEOUT_QUIET=true
GRUB_TIMEOUT=10
GRUB_DISTRIBUTOR=`lsb_release -i -s 2> /dev/null || echo Debian `
GRUB_CMDLINE_LINUX_DEFAULT="quiet splash"
GRUB_CMDLINE_LINUX=""
# GRUB_TERMINAL=console
# GRUB_GFXMODE=640x480
# GRUB_DISABLE_LINUX_UUID=true
# GRUB_DISABLE_RECOVERY="true"
# GRUB_INIT_TUNE="480 440 1"
```

Стандартная конфигурация в Fedora выглядит так:

```
GRUB_TIMEOUT=5
GRUB_DISTRIBUTOR="Fedora"
GRUB_DEFAULT=saved
GRUB_CMDLINE_LINUX="rd.md=0 rd.dm=0 quiet SYSFONT=latarcyrheb-sun16 rhgb \
   rd.lvm.lv=vg_fedora16vbox/lv_root rd.luks=0 LANG=de_DE.UTF-8 \
   rd.lvm.lv=vg_fedora16vbox/lv_swap KEYTABLE=de-latin1-nodeadkeys"
```

Далее объясняются значения многочисленных параметров.

- Переменная `GRUB_DEFAULT` указывает, какая запись меню GRUB должна выбираться по умолчанию. Обратите внимание на то, что количество записей может изменяться после каждого вызова `update-grub` и особенно после каждого обновления ядра. Поэтому такая настройка, как `GRUB_DEFAULT=5`, обычно не подходит.

Настройка saved означает, что будет активизирована запись меню, которая использовалась последней. Правда, это работает лишь при условии, что файлы GRUB расположены в обычном разделе диска! Если же используется LVM или RAID, то после выбора из меню GRUB не сможет сохранять каких-либо переменных окружения.

Вы также можете присвоить GRUB_DEFAULT последовательность символов (строку) menuentry, где menuentry — желаемая строка меню. Но при этом строго соблюдайте порядок записи данной строки (в том числе все номера версий и устройств, которые могут находиться в записи меню).

- Переменная GRUB_HIDDEN_TIMEOUT важна в тех случаях, когда при установке GRUB опознает на вашем компьютере только одну операционную систему (в нашем случае — Linux). В таком случае значение GRUB_HIDDEN_TIMEOUT задает, сколько времени будет у пользователя, чтобы отобразить меню GRUB нажатием клавиши Shift (в течение этого времени ожидания экран остается черным).

  В Ubuntu значение параметра GRUB_HIDDEN_TIMEOUT=0 приводит к тому, что GRUB сразу запускает операционную систему. Чтобы вмешаться в процесс загрузки, нужно нажать Shift сразу же после запуска компьютера. Если на компьютере установлено несколько операционных систем, то GRUB игнорирует настройку GRUB_HIDDEN_TIMEOUT и сразу отображает меню.

- Переменная GRUB_HIDDEN_TIMEOUT_QUIET=true отменяет показ счетчика с обратным отсчетом в период ожидания, заданный в GRUB_HIDDEN_TIMEOUT.

  Если вы хотите, чтобы меню GRUB отображалось (в том числе и тогда, когда на компьютере установлена только Ubuntu), закомментируйте строки GRUB_HIDDEN_TIMEOUT=... и GRUB_HIDDEN-TIMEOUT_QUIET=... (поставьте перед каждой из них символ комментария #).

- Переменная GRUB_TIMEOUT=n указывает, в течение какого количества секунд GRUB дожидается выбора одной из записей меню. Если за это время не поступит никакого пользовательского ввода, GRUB запустит уже выбранную операционную систему. Указанное здесь время учитывается лишь в том случае, если меню GRUB вообще появляется на экране, то есть, если на компьютере установлено несколько операционных систем или пользователь нажал клавишу Shift во время периода ожидания, заданного в GRUB_HIDDEN_TIMEOUT.

- Переменная GRUB_DISTRIBUTOR интерпретируется сценарием 10_linux (см. ниже), в ней содержится имя текущего дистрибутива (например, Fedora или Ubuntu).

- Значения переменных GRUB_CMDLINE_LINUX и GRUB_CMDLINE_LINUX_DEFAULT также учитываются сценарием 10_linux. В них указывается, какие параметры должны быть сообщены ядру. Параметры GRUB_ CMDLINE_LINUX действуют при каждом запуске, а параметры GRUB_CMDLINE_LINUX_DEFAULT дополнительно применяются при запуске, происходящем по умолчанию (но не в режиме восстановления).

- По умолчанию меню GRUB отображается в графическом режиме с разрешением 640 × 480 пикселов. Если вы хотите работать с более высоким разрешением, его можно задать в переменной GRUB_ GFXMODE (разумеется, в пределах возможностей вашей графической карты). Если вы желаете вообще отказаться от

графического режима, активизируйте специально предусмотренную для этого настройку GRUB_TERMINAL=console. Обе переменные интерпретируются сценарием 00_header. При стандартных настройках значительные различия между текстовым и графическим режимами отсутствуют, но только в графическом режиме могут отображаться символы Unicode.

- Как правило, GRUB передает ядру, которое предполагается запустить, корневой каталог в качестве UUID-номера. Если вы, напротив, предпочитаете указывать номер устройства (например, /dev/sda1), активизируйте строку GRUB_DISABLE_LINUX_UUID=true. Эта настройка влияет только на запуск активного дистрибутива (сценарий 10_linux), но не на другие дистрибутивы.
- update-grub или grub-mkconfig создают, как правило, записи меню и для запуска Linux в режиме восстановления. При этом Linux запускается в однопользовательском режиме, без отображения экрана-заставки. Если вы хотите отказаться от записей, необходимых для восстановления, активизируйте строку GRUB_DISABLE_RECOVERY="true".
- С помощью GRUB_INIT_TUNE= можно сопроводить запуск GRUB звуковым сигналом.

Автоматическое создание файла grub.cfg. Основная идея конфигурации GRUB заключается в том, что сами вы задаете лишь ключевые данные этой конфигурации. Готовый конфигурационный файл grub.cfg создается специальным сценарием с учетом указанных вами параметров. Здесь важно подчеркнуть, что сценарии GRUB пытаются опознать *все* операционные системы, установленные на компьютере (Linux, Windows и т. д.) и сделать в файле grub.cfg соответствующие записи. Это же касается и тех операционных систем, которые могут быть позже добавлены на компьютер.

Для этого в каталоге /etc/grub.d содержатся исполняемые скриптовые (сценарные) файлы (табл. 15.1). Когда следует создать новую версию grub.cfg, программа update-grub по порядку выполняет все сценарии, содержащиеся в grub.d. Результат (стандартный вывод) сценариев сохраняется в grub.cfg. Но из-за такого метода работы конфигурационные файлы получаются довольно путаными: в коде то и дело встречаются команды вида cat " EOF, переадресующие в стандартный вывод все строки вплоть до аббревиатуры EOF. Сами эти строки часто содержат сценарный код, который интерпретируется GRUB только при запуске системы.

Таблица 15.1. Файлы из каталога /etc/grub.d/

Файл	Функция
00_header	Основные настройки GRUB
05_debian_theme	Цветовое оформление меню (только в Debian и Ubuntu)
10_linux	Записи меню для запуска актуального дистрибутива
20_linux_xen	Записи меню для запуска виртуальных машин
20_memtest86+	Запись меню для запуска Memtest86 (только в Debian и Ubuntu)
30_os-prober	Записи меню для запуска других операционных систем
40_custom	Шаблон для собственных конфигурационных файлов
41_custom	Добавление в grub.cfg код из custom.cfg

Два наиболее интересных сценария — 10_linux и 30_os-prober. Сценарий 10_linux выдает для каждой версии ядра, находящейся в каталоге /boot/, две записи меню: одну для обычного запуска и одну для восстановительного запуска в однопользовательском режиме, без экрана-заставки. Записи меню сортируются по номеру версии, причем в начале списка идет самая новая версия.

30_os-prober вызывает сценарий os-prober. Этот сценарий выводит список всех операционных систем (Linux, Windows, Mac OS X), расположенных на всех доступных сегментах жесткого диска. Для каждой из этих операционных систем создается своя запись в меню, причем, в случае с дистрибутивами Linux, может быть применен вариант конфигурации GRUB, который уже задан. При этом вызываются многочисленные сценарии, входящие в состав пакета os-prober.

Во время моих тестов распознавание различных дистрибутивов Linux и версий Windows на компьютерах BIOS функционировало великолепно. С EFI все было гораздо сложнее. Операционные системы, запускаемые через EFI, *не находят* входа в меню GRUB. В общем-то, в этом нет ничего удивительного: вы же можете выбрать операционную систему в меню EFI, зачем же GRUB решать ту же задачу? Но, по моему опыту, интеграция с меню GRUB все же является желательной. На многих компьютерах меню EFI отображается именно в тех случаях, когда при запуске компьютера в определенный момент времени нажато нужное сочетание клавиш. Я об этом не раз забывал и оказывался в операционной системе, в которую совершенно не собирался попадать! В автоматическом меню GRUB ошибку допустить намного сложнее.

Самостоятельное внесение дополнений в grub.cfg. Если вы хотите дополнить grub.cfg собственными записями, добавьте в каталог grub.d свои сценарии. Эти сценарии выполняются в том порядке, который определяется их нумерацией (при одинаковом порядковом номере — в алфавитном порядке). Не забудьте поставить бит исполнения (Execute-Bit). Обратите внимание и на то, что в grub.cfg попадает не сам файл, а его результат (стандартный вывод)!

В файле шаблона 40_custom показан пример подобной работы. Здесь к файлу применяется команда tail (параметр $0). Благодаря параметру -n +3, файл выводится, начиная с третьей строки, первые две строки не отображаются.

```
#!/bin/sh
exec tail -n +3 $0
# в этом файле показан пример добавления собственных записей
...
```

В другом варианте показан пример 41_custom. Содержащийся здесь код без изменений переносится в grub.cfg и интерпретируется только к моменту запуска (то есть при отображении меню GRUB). В этот момент проверяется, существует ли файл /boot/grub/custom.cfg (Ubuntu) или /boot/grub2/custom.cfg (Fedora, openSUSE). Если он существует, то этот файл добавляется в конфигурацию GRUB. Поскольку source исполняется при выполнении GRUB, изменения в custom.cfg вступают в силу и не приходится дополнительно создавать конфигурационный файл GRUB grub.cfg.

```
#!/bin/sh
cat <<EOF
if [ -f \$prefix/custom.cfg ]; then
```

```
   source \$prefix/custom.cfg;
fi
EOF
```

Синтаксис и внутренняя организация

Размеры книги не позволяют привести здесь полное описание синтаксиса всех ключевых слов, допустимых для использования в `grub.cfg`. Поэтому ниже я расскажу лишь о важнейших ключевых словах, которые употребляются в примерах из этого раздела. Официальный учебник по GRUB (PDF-документ содержит более 100 страниц!) находится по адресу http://www.gnu.org/software/grub/manual/.

Переменные. С помощью `set varname=value` осуществляется присваивание переменных. Для считывания переменных применяется запись вида `$varname`. При интерактивном выполнении команд GRUB команда `echo $varname` отображает содержимое переменной, а `set` возвращает все определенные переменные.

Некоторые переменные обладают, кроме того, особым значением. К ним относятся, например, `default`, `timeout`, `color_xxx`, `menu_color_xxx` и особенно `root`: при любом доступе к файлам автоматически происходит считывание раздела, определенного в `root`.

GRUB может долговременно сохранять переменные на время работы. Для этого сначала (в Linux) должен быть создан файл `/boot/grub[2]/grub-editenv` (по умолчанию в большинстве дистрибутивов он уже создан).

```
root# grub-editenv /boot/grub[2]/grubenv create
```

Теперь во время работы GRUB может с помощью `save_env` *имяпеременной* сохранить в этом файле переменную либо считывать переменные из этого файла с помощью `load_env`. Перед этим необходимо настроить `root` так, чтобы эта переменная указывала на раздел с файлом окружения. В Linux вы также можете считывать или изменять переменные GRUB с помощью `grub-editenv`.

Разделы. В GRUB существует собственная номенклатура для обозначения жестких дисков и содержащихся на них разделов (табл. 15.2). Но нумерация при этом не совпадает: первый жесткий диск имеет номер 0, а первый раздел — номер 1!

Таблица 15.2. Имена разделов в GRUB 2

Имя **устройства GRUB 2**	**Значение**
(hd0)	Первый жесткий/твердотельный диск (соответствует /dev/sda)
(hd1)	Второй жесткий/твердотельный диск (соответствует /dev/sdb)
(hd0,1)	Первый раздел первого жесткого/твердотельного диска (/dev/sda1)
(hd2,8)	Восьмой раздел второго жесткого/твердотельного диска (/dev/sdc8)

Перед номером раздела допускается ставить сокращенное обозначение, указывающее способ секционирования: `msdos` для носителей, чья таблица секционирования записана в главной учетной записи, `gpt` — для носителей, применяющих таблицу разбиения GUID. Таким образом могут получаться названия вида `(hd0,msdos3)` или `(hd0,gpt2)`.

СОВЕТ

На жестких дисках SATA и твердотельных дисках в принципе совершенно не важно, с какими SATA-портами материнской платы связаны носители данных. Первым жестким/твердотельным

диском с точки зрения GRUB является тот, который связан с SATA-портом, имеющим наименьший номер.

Мой опыт, так или иначе, подсказывает, что рекомендуется связывать первый жесткий или твердотельный диск именно с первым портом SATA. При несоблюдении этого условия во время моих тестов время от времени GRUB опознавал жесткий диск только как hd1. В таком случае автоматически сгенерированный конфигурационный файл GRUB не функционировал. Чтобы загрузиться в таком случае, нужно отредактировать соответствующую запись GRUB: нажать клавишу E и заменить hd0,... на hd1,...

Работа с номерами UUID. В `grub.cfg` часто встречается следующая последовательность команд:

```
set root=(hd1,1)
search --no-floppy --fs-uuid --set 12345678...
```

Первая команда инициализирует переменную `root`. Вторая команда ищет файловую систему с указанным номером UUID. Если поиск оканчивается успехом, то GRUB, на основании значения параметра `--set`, сохраняет соответствующее имя раздела в переменной `root`.

Такая двоякость допускается в качестве меры предосторожности. Так мы можем гарантировать, что GRUB найдет раздел диска и в том случае, если перед этим он был заново отформатирован (изменился номер UUID) либо если номер носителя данных (например, флешки) изменился из-за переподключения устройств.

Модули. С помощью `insmod name` GRUB загружает во время работы модули расширений с дополнительными функциями. GRUB ищет файлы модулей `name.mod` в каталоге `/boot/grub[2]`, в разделе, указанном в переменной `root`. Следует в частности отметить важные модули `part_msdos` и `part_gpt` (считывание таблиц секционирования), `ext2` (файловые системы ext2 и ext4), `raid`, `raid5rec`, `raid6rec` и `mdraid` (программные массивы RAID), `lvm`, `gfxterm` (графическая консоль), `vbe` (графическая система), а также `jpeg`, `tga` и `png` для считывания графических файлов.

Записи меню GRUB

Menuentry. Записи меню GRUB вводятся ключевым словом `menuentry`. Следующий за ним текст записывается в кавычках, поэтому в нем допускаются интернациональные символы.

ВНИМАНИЕ

Записи меню, содержащие синтаксические ошибки, GRUB во время работы просто игнорирует и не отображает. При этом не выводится никаких сообщений об ошибках, из-за чего находить такие ошибки очень сложно. Лучше всего перейти в работающем GRUB в командный режим работы (с помощью клавиши C) и ввести команду, соответствующую записи меню.

Подменю. В версии GRUB 2.00 и выше поддерживаются подменю (меню второго порядка). В файле `grub.cfg` подменю определяются так:

```
submenu 'Подменю' {
   menuentry 'Запись 1' { ... }
   menuentry 'Запись 2' { ... }
}
```

В некоторых дистрибутивах в подменю стали переноситься записи для старых версий ядра, а также для восстановления системы. Благодаря этому основное меню становится значительно нагляднее. При работе с GRUB для перехода из подменю в меню первого порядка нажмите клавишу Esc.

Запуск Linux. Запись в меню GRUB 2, предназначенная для запуска Linux, в минимальном варианте выглядит примерно так:

```
menuentry "Linux" {
   set root=(hd0,3)
   linux /boot/vmlinuz-n.n.n root=... ro quiet splash
   initrd /boot/initrd.img-n.n.n
}
```

`set root` указывает раздел, в котором находится ядро и Initrd-файл. Ключевые слова `linux` и `initrd` задают имена файлов относительно раздела диска. Указанные параметры передаются ядру. Обязательно требуются `root` для указания системного раздела и `ro`, чтобы доступ к системному разделу был открыт сразу только для чтения. Все остальные параметры специфичны для конкретных дистрибутивов. Вот примеры:

- Debian 6 — `root=/dev/xxx ro quiet`;
- Fedora 17 — `root=/dev/xxx ro rhgb quiet LANG=... SYSFONT=... KEYTABLE=...`;
- openSUSE 12.2 — `root=UUID=xxx resume=/dev/xxx splash=silent showopts video=nxn quiet`;
- Ubuntu 12.10 — `root=UUID=xxx ro quiet splash`.

Если в системе есть отдельный загрузочный раздел, то укажите его с помощью `set root`. В таком случае в `linux` и `initrd` уже не требуется задавать загрузочный каталог:

```
menuentry "Linux - Mit eigener Bootpartition" {
   set root=(hd0,2)
   linux /vmlinuz-n.n.n  root=... ro quiet splash
   initrd /initrd.img-n.n.n
}
```

Если раздел с ядром и Initrd-файлом входит в состав LVM-системы и/или программного массива RAID, то для работы потребуется загрузить соответствующие модули GRUB. В случае с RAID-5 и, соответственно, RAID-6 применяются модули `raid5rec` и `raid6rec`. В `set root` теперь можно указать системный раздел с помощью, соответственно, `(`*`lvname`*`)` и `(md`*`n`*`)`:

```
menuentry "Linux - Mit Software-RAID" {
   insmod raid mdraid
   set root=(md0)
   linux /boot/vmlinuz-n.n.nroot=... ro quiet splash
   initrd /boot/initrd.img-n.n.n
}
menuentry "Linux - Mit LVM" {
   insmod lvm
   set root=(vg1-root)
   linux /boot/vmlinuz-n.n.nroot=... ro quiet splash
```

```
    initrd /boot/initrd.img-n.n.n
}
menuentry "Linux - LVM auf RAID-5" {
    insmod raid raid5rec mdraid lvm
    set root=(vg1-root)
    linux /boot/vmlinuz-n.n.nroot=/dev/mapper/... ro quiet splash
    initrd /boot/initrd.img-n.n.n
}
```

Если вы не хотите полагаться на номера устройств, вы можете также сами найти системный раздел в таблице UUID с помощью команды `search`. Если команда `search` сработает успешно, то GRUB-переменная `root` будет соответствующим образом изменена. Это работает и с LVM-, и с RAID-разделами, если предварительно загрузить нужные модули GRUB. В Ubuntu по умолчанию за `set root` следует подходящая команда `search`, позволяющая максимально снизить зависимость GRUB от номеров устройств.

```
menuentry " - настройка root-переменной Linux по  UUID " {
    set root=(hd0,3)
    search --no-floppy --fs-uuid --set 12345678...
    linux /boot/vmlinuz-n.n.nroot=... ro quiet splash
    initrd /boot/initrd.img-n.n.n
}
```

Запуск Windows. Для запуска Windows на компьютере с BIOS выберите с помощью `set root` системный раздел с Windows и запустите его загрузчик, используя `chainloader +1`. Обратите внимание на то, что Windows версии 7 и выше по умолчанию создает на диске два раздела: один загрузочный раздел размером около 100 Мбайт с файлами `bootmgr` и `bootsect.bak` и другой, значительно более крупный, системный раздел. В GRUB требуется указывать загрузочный раздел. Команда `search` остается опциональной. От `drivemap`, как правило, можно отказаться. Команда пытается «уверить» Windows в том, что система Windows расположена на первом жестком диске, даже если это совсем не так. Иногда требуется в первую очередь запускать именно Windows.

```
menuentry "Windows 7" {
    set root=(hd0,1)
    search --no-floppy --fs-uuid --set 12345678...
    drivemap -s (hd0) $root
    chainloader +1
}
```

О том, как запустить Windows на компьютере с EFI с помощью GRUB, рассказано в следующем разделе.

Ветвление и переход к работе с другим загрузчиком

Уже упоминавшаяся выше команда GRUB `chainloader` позволяет путем ветвления перейти к работе с другим загрузчиком. Соответствующие команды лучше всего

вставить в конфигурационный файл, оформляемый по образцу /etc/grub.d/40_custom.

Запуск загрузчика из другого раздела диска. Если вы установите на компьютере еще один загрузчик в стартовом секторе раздела, то в меню GRUB 2 можно перейти к работе с этим загрузчиком. В таком случае укажите этот раздел с помощью set root (или search) и выполните chainloader +1:

```
menuentry "GRUB in /dev/sdb7" {
   set root=(hd1,7)
   search --no-floppy --fs-uuid --set 12345678...
   chainloader +1
}
```

В данном случае у меня возникли проблемы, когда я пытался ветвлением перейти в другой установленный экземпляр GRUB. Система выдавала сообщение об ошибке invalid signature (недействительная подпись). Частично справиться с этой проблемой удавалось, дополнительно указав параметр --force:

```
menuentry "GRUB2 in /dev/sdb7" {
   set root=(hd1,7)
   chainloader +1 --force
}
```

К сожалению, это тоже работает не всегда. Особенно сложно переходить с GRUB 2 на GRUB 0.97. В этом случае целесообразно вообще отказаться от set root и задать желаемый раздел непосредственно с помощью chainloader:

```
menuentry "GRUB 0.97 in /dev/sdb7" {
   chainloader (hd1,7)+1 --force
}
```

Если в другом дистрибутиве также используется GRUB 2, то ключевое слово config-file позволяет загрузить конфигурационный файл GRUB grub.cfg этого дистрибутива:

```
menuentry "GRUB in /dev/sdb7" {
   set root=(hd1,7)
   search --no-floppy --fs-uuid --set 12345678...
   configfile /boot/grub/grub.cfg
}
```

ВНИМАНИЕ

На компьютерах с BIOS GRUB 2 предназначен только для записи в главную установочную запись (MBR) жесткого или твердотельного диска! В GRUB 0.97 нередко практиковалась установка GRUB в стартовый сектор системного раздела. В учебнике по GRUB 2 делать это категорически не рекомендуется!

Ветвление для работы с загрузчиком EFI. На компьютерах с EFI можно прямо из GRUB запускать другой загрузчик EFI. В приведенных далее примерах предполагается, что раздел с EFI является первым разделом первого жесткого диска; если это не так, то нужно соответствующим образом откорректировать set root.

```
menuentry " Запуск загрузчика Windows (EFI)" {
   insmod part_gpt
   set root='(hd0,1)'
   chainloader /EFI/Microsoft/Boot/bootmgfw.efi
}

menuentry "Запуск загрузчика Fedora (EFI)" {
   insmod part_gpt
   set root='(hd0,1)'
   chainloader /EFI/redhat/grub.efi
}

menuentry "Запуск загрузчика Ubuntu (EFI)" {
   insmod part_gpt
   set root='(hd0,1)'
   chainloader /EFI/ubuntu/grubx64.efi
}
```

Чтобы быстро узнать, какой загрузчик EFI вам подходит, выполните следующую команду:

```
root# find /boot/efi -name '*.efi' | sort
/boot/efi/EFI/Boot/bootx64.efi
/boot/efi/EFI/Microsoft/Boot/bootmgfw.efi
/boot/efi/EFI/Microsoft/Boot/bootmgr.efi
/boot/efi/EFI/Microsoft/Boot/memtest.efi
/boot/efi/EFI/redhat/grub.efi
/boot/efi/EFI/ubuntu/grubx64.efi
```

ПРИМЕЧАНИЕ

В системах Windows в файле bootmgfw.efi содержится загрузчик. Но сам файл bootmgr.efi не запускается. Запустить memtest.efi мне также не удалось. Судя по сообщениям, которые мне удалось найти на форумах, ветвление к другому EFI-загрузчику срабатывает не на каждом компьютере. Но на моем тестовом компьютере с материнской платой ASUS P8H67-M Evo никаких проблем не возникло.

Индивидуальная конфигурация

GRUB 2 заранее сконфигурирован так, чтобы любые операционные системы можно было запускать на компьютере с BIOS. Стандартная конфигурация функционирует хорошо и, как правило, в большинстве случаев ее достаточно. Этот подраздел ориентирован только на тех пользователей Linux, которые хотят настроить GRUB под себя.

Деактивация os-probe. Если на жестком диске компьютера много разделов, то выполнение сценария `30_os-probe` длится достаточно долго. Если данный сценарий вам не нужен (например, для запуска других дистрибутивов вы предпочитаете использовать самостоятельно определенные записи в меню GRUB), то добавьте в `/etc/default/grub` строку `GRUB_DISABLE_OS_PROBER=true`.

Фоновая графика. В качестве фона меню GRUB можно задать картинку. GRUB может работать с форматами JPG, PNG и TGA. Код, необходимый для интеграции

фоновой картинки, в Debian и Ubuntu уже содержится в сценарии 05_debian_theme. Нужно всего лишь изменить три переменные и задать имя файла с вашим изображением, а также желаемые цвета для текста (рис. 15.2).

```
# в /etc/grub.d/05_debian_theme
...
WALLPAPER="/boot/grub/myown.png"
COLOR_NORMAL="white/black"
COLOR_HIGHLIGHT="yellow/black"
```

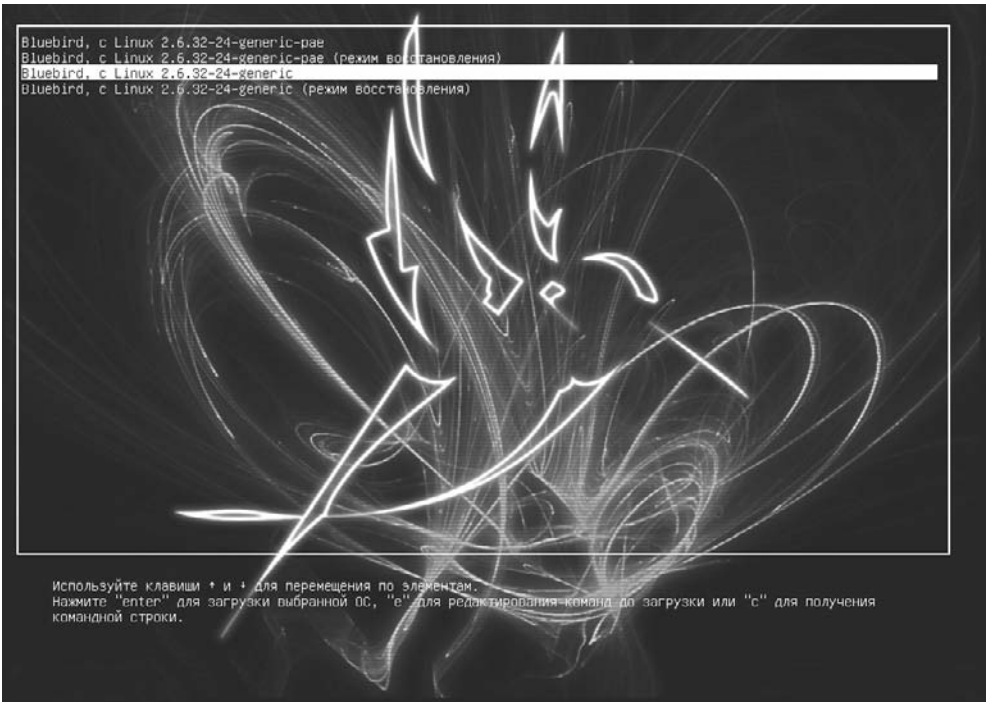


Рис. 15.2. GRUB 2 с индивидуально выбранным фоном

Закрепление операционной системы по умолчанию. В файле /etc/default/grub можно закрепить номер определенной записи в меню GRUB. Операционная система, указанная под этой записью, будет запускаться автоматически. На практике пользы от этого немного. Если вы, например, хотите, чтобы по умолчанию запускалась система Windows XP, а в меню GRUB запись с Windows XP идет на десятом месте, то задайте параметр GRUB_DEFAULT=9 (отсчет начинается с нуля). Но не исключено, что после обновления ядра в меню GRUB появятся две дополнительные записи и ваша настройка станет неверной. Гораздо удобнее оставить параметр GRUB_DEFAULT=0, а желаемую запись меню GRUB разместить перед всеми остальными. Для этого лучше всего воспользоваться дополнительным сценарием в /etc/grub.d, причем имя файла должно начинаться с целого числа меньше 10. Возьмем следующие строки в качестве образца:

```
#!/bin/sh
exec tail -n +3 $0
# Файл /etc/grub.d/09_boot-windows-by-default
menuentry "Windows 7" {
   set root=(hd0,1)
   chainloader +1
}
```

Другая возможность заключается в том, чтобы точно задать запись в меню GRUB, например, вот так:

```
GRUB_DEFAULT='Windows 7 (loader) (on /dev/sda1)'
```

Но такой метод целесообразен лишь при условии, что при обновлениях GRUB номер этой записи в меню не изменяется. С Windows это работает хорошо, чего не скажешь о Linux, поскольку в автоматически генерируемых записях меню указана версия ядра — а она меняется при каждом обновлении ядра.

Установка вручную и первая помощь при работе с компьютерами с BIOS

Как правило, GRUB 2 правильно создается при установке вашего дистрибутива Linux. В дальнейшем вы, возможно, будете вносить изменения в конфигурацию GRUB, то есть в файл `grub.cfg`, но установкой GRUB как таковой заниматься больше не придется.

Этот подраздел о компьютерах с BIOS и следующий подобный подраздел о компьютерах с EFI актуален лишь в случаях, когда вы по каким-то причинам устанавливаете GRUB вручную либо когда хотите исправить поврежденную установку GRUB (например, если другая операционная система перезаписывает информацию в MBR).

Установка вручную с помощью grub-install. Сценарий `grub-install` (в Fedora и openSUSE он называется `grub2-install`) устанавливает загрузчик в первый сектор указанного жесткого или твердотельного диска (то есть в MBR и во все остальные сектора, которые находятся на диске до начала первого раздела). Как правило, в качестве единственного параметра вы передаете в данном случае имя устройства, соответствующее носителю данных. При этом допускается запись как по принципу Linux (например, `/dev/sda`), так и по принципу GRUB (например, `(hd0)`):

```
root# grub-install /dev/sda
```

Теоретически возможно выполнить установку и в стартовом секторе сегмента (это записывалось бы как `grub-install/dev/sda3` или `(hd0,3)`), но этого не рекомендуется делать, если вы не работаете с GRUB 0.97. В стартовом секторе слишком мало места для внедрения всего кода GRUB, а при применении блок-списков система считается нестабильной. (Блок-список содержит информацию о том, в каких блоках с данными на жестком/твердотельном диске находится следующая часть GRUB-кода.) Если эти аргументы кажутся вам неубедительными, можете произвести принудительную установку с помощью параметра `--force`.

Особый случай: ПК с BIOS и GPT. Linux можно инсталлировать на жестком/твердотельном диске, на котором находится таблица секционирования GUID, а сам диск установлен на компьютере с BIOS. (Что касается Windows, то эта операционная система, напротив, может быть установлена только на ПК с EFI, если на жестком или твердотельном диске содержится таблица секционирования GPT.) Так или иначе, в подобном особом случае в учебнике по GRUB рекомендуется выделять для установки GRUB собственный раздел размером около 1 Мбайт, снабженный флагом `bios_grub`. Этот раздел предусмотрен только для установки BIOS-совместимых загрузчиков.

Такой раздел `bios_grub` не нуждается в форматировании. До сих пор Fedora 16 остается единственным дистрибутивом, самостоятельно создающим такой раздел при автоматическом сегментировании жестких дисков с таблицей GPT (в версии 17 Fedora снова отказалась от того, чтобы автоматически создавать GPT на пустых жестких дисках).

Разумеется, раздел диска можно создать и вручную. Для этого поставьте флаг `bios_grub` с `parted`, заменив *n* на желаемый номер раздела:

```
root# parted /dev/sda set n bios_grub on
```

ВНИМАНИЕ

Ни в коем случае не помечайте флагом bios_grub раздел, в котором находятся данные! При установке GRUB начало раздела переписывается; файловая система, которая может находиться в разделе, в таком случае оказывается необратимо испорченной!

Когда при установке GRUB обнаруживает наличие раздела `bios_grub`, этот загрузчик, как обычно, устанавливает первую часть кода GRUB в главную загрузочную запись, а остальной код — в раздел, помеченный как `bios_grub`. Применять именно такой подход при установке GRUB на жестких дисках с GPT, конечно, необязательно. Но этот подход считается достаточно надежным вариантом, особенно если на компьютере установлены другие операционные системы (Windows). Но у этого метода есть и свой недостаток: если на компьютере существует такой раздел, то на машине становится невозможно параллельно инсталлировать несколько экземпляров GRUB-2, поскольку при каждой новой установке GRUB содержимое раздела `bios_grub` перезаписывается. Более подробно о GRUB и GPT рассказано здесь: http://www.gnu.org/software/grub/manual/html_node/BIOS-installation.html и http://www.wensley.org.uk/gpt.

Ремонт GRUB с помощью «живого диска» (на компьютерах с BIOS). Если при установке GRUB что-то пошло не так либо загрузчик был стерт при установке другой операционной системы, то GRUB нужно переустановить с «живого диска», на котором находятся инструменты с актуальными инструментами GRUB 2. После запуска системы перейдите в режим администратора (root) (в Ubuntu — `sudo -s`), подключите системный раздел, а также активные каталоги `/dev`, `/proc` и `/sys` к файловой системе, а потом выполните `chroot`. Возможно, после этого понадобится также подключить загрузочный раздел к новой root-файловой системе. Затем обновите конфигурацию GRUB и запишите GRUB в желаемое место с помощью `grub[2]-install` (обычно для этого используется главная загрузочная запись первого жесткого диска). Как обычно, в следующих командах необходимо заменить `/dev/sda`*n* вашими номерами устройств.

```
root# mkdir /syspart
root# mount /dev/sda2 /syspart     (Системный раздел)
root# mount -o bind /dev  /syspart/dev
root# mount -o bind /proc /syspart/proc
root# mount -o bind /sys  /syspart/sys
root# chroot /syspart
root# mount /dev/sda1 /boot        (Загрузочный раздел при его наличии)
root# update-grub                  (Ubuntu)
root# grub-install /dev/sda        (Ubuntu)
root# grub2-mkconfig -o /boot/grub2/grub.cfg   (Fedora, openSUSE)
root# grub2-install /dev/sda       (Fedora, openSUSE)
root# exit
```

Установка вручную и первая помощь при работе с компьютерами с EFI

Этот подраздел посвящен исключительно работе с Ubuntu, поскольку по состоянию на середину 2012 года данный дистрибутив был единственным, который применял GRUB 2 на компьютерах с EFI. Другие протестированные мной дистрибутивы на тот момент были либо вообще несовместимы с EFI, либо использовали другие загрузчики.

Обратите внимание: при EFI-установке дистрибутива Fedora 17 вместе с дистрибутивом инсталлируется EFI-совместимая версия GRUB 0.97. Конечно, потом можно дополнительно установить EFI-совместимую версию GRUB 2, но мои попытки настроить вручную GRUB 2 после такой дополнительной инсталляции ничем не увенчались. Очевидно, в пакете Fedora `grub2-efi` есть ошибка, о чем свидетельствует и следующее сообщение: https://bugzilla.redhat.com/show_bug.cgi?id=816742.

Установка вручную с помощью grub-install. Установить вручную GRUB 2 на компьютере с EFI довольно несложно. Не приходится сообщать `grub-install` никаких параметров:

```
root# grub-install
```

Эта команда создает каталог `/boot/efi/EFI/`*`distributionsname`*. В этот каталог записывается новый загрузочный файл с расширением `.efi`, содержащий GRUB-код. Кроме того, файл `.efi` добавляется в число загрузочных записей EFI, в этом списке он располагается на первом месте. При этом `grub-install` обращается к команде `efibootmgr`, о которой мы подробнее поговорим далее.

Для успешного выполнения `grub-install` должны соблюдаться несколько предварительных условий.

- Конфигурация GRUB должна быть подготовлена в файле `/boot/grub[2]/grub.cfg`.
- EFI-раздел должен быть подключен к дереву каталогов через путь `/boot/efi`.
- Должна быть установлена команда `efibootmgr` из одноименного пакета.
- Должен быть загружен модуль ядра `efivars`. Но `modprobe efivars` удается лишь в том случае, когда дистрибутив был запущен в EFI-режиме, а не в BIOS-режиме. Поэтому без EFI-совместимого «живого диска» сложно перестроить на работу

с EFI дистрибутив Linux, который первоначально устанавливался для функционирования с BIOS.

Ремонт GRUB с помощью «живого диска» (на компьютерах с EFI). Исправление ошибок GRUB в режиме EFI происходит примерно так же, как и в случае с BIOS (см. выше). Но при этом исключительно важно то, что «живой диск» должен быть запущен в режиме EFI, а не в режиме BIOS.

После запуска системы перейдите в режим администратора (root), подключите системный раздел, а также активные каталоги `/dev`, `/proc` и `/sys` к файловой системе, а потом выполните `chroot`. Возможно, после этого понадобится еще и подключить загрузочный раздел к новой root-файловой системе. Затем обновите конфигурацию GRUB и запишите GRUB в раздел EFI с помощью `grub-install`. Как обычно, в следующих командах необходимо заменить `/dev/sda`*n* вашими собственными номерами устройств.

```
root# mkdir /syspart
root# mount /dev/sda2 /syspart          (Системный раздел)
root# mount -o bind /dev  /syspart/dev
root# mount -o bind /proc /syspart/proc
root# mount -o bind /sys  /syspart/sys
root# chroot /syspart
root# mount /dev/sda1 /boot/efi         (Раздел с EFI)
root# update-grub
root# grub-install
root# exit
```

Вручную вводим команды GRUB для запуска Linux

Если GRUB удается запустить, но после выбора Linux из меню GRUB саму операционную систему запустить не получается, то можно, нажав клавишу C, перейти в меню GRUB в интерактивный режим, а затем выполнить следующие команды:

```
grub> set root=(hd0,1)
grub> linux /vmlinuz root=/dev/sda1
grub> initrd /initrd.img
grub> boot
```

Вместо `(hd0,1)` и `/dev/sda1` задайте имя вашего системного раздела с Linux. В большинстве дистрибутивов файлы `/vmlinuz` и `/initrd.img` указывают на новейшую версию ядра и файла Initrd. Эти файлы находятся в каталоге `/boot`. Если в вашем случае это не так, нужно точно задать местоположение ядра и Initrd-файла. GRUB обеспечивает при этом функцию автозавершения, оно выполняется клавишей Tab.

Изменение загрузочных записей и настроек EFI вручную (efibootmgr)

Откуда EFI известно, какие операционные системы установлены на компьютере и, соответственно, какие загрузочные записи должны отображаться? На материнских платах с EFI эта информация хранится в энергонезависимой памяти (NVRAM).

Всякий раз при установке новой операционной системы, после создания *.efi-файла в EFI-разделе, соответствующая запись сохраняется в энергонезависимой памяти.

В Linux можно считывать и изменять EFI-данные, сохраненные таким образом, с помощью команды efibootmgr. Данная команда предполагает, что в системе загружен модуль ядра efivars. Если это не так, то выполните modprobe efivars. Обратите внимание: модуль efivars можно применять лишь в том случае, когда Linux был загружен в режиме EFI (а не в режиме BIOS!). Для восстановительных работ используйте по возможности «живой диск» с Linux, который можно запускать в EFI-режиме.

Если команда efibootmgr выполняется без дополнительных параметров, то она выводит список загрузочных записей EFI, а также некоторых других параметров загрузчика EFI. Следующий вывод, например, означает, что на компьютере установлены Ubuntu, Fedora и Windows в EFI-режиме. При перезапуске по истечении времени ожидания (около одной секунды) автоматически запустится Ubuntu (BootCurrent). В период ожидания можно отобразить меню EFI с помощью сочетания клавиш, специфичного для конкретной материнской платы (на моем компьютере требуется нажать F8).

```
root# efibootmgr
BootCurrent: 0000
Timeout: 1 seconds
BootOrder: 0000,0005,0003,0001,0002
Boot0000* ubuntu
Boot0001* Hard Drive
Boot0002* CD/DVD-drive
Boot0003* Windows Boot Manager
Boot0005* Fedora
```

Если вы хотите после перезапуска один раз запустить Fedora, отметьте загрузочную запись этого дистрибутива параметром -n:

```
root# efibootmgr -n 5
```

Если же порядок загрузки потребуется изменить надолго, то сопроводите желаемую загрузочную запись параметром -o:

```
root# efibootmgr -o 5
```

Следующая команда создает новую загрузочную запись EFI. Путь задается относительно раздела с EFI (то есть /boot/efi), при этом должен использоваться разделитель каталогов \. Удваивать символ \ необходимо потому, что оболочка интерпретирует одиночный слэш \ как указатель специальных символов. С помощью параметра -L можно указать имя, которое должно отображаться в меню EFI.

```
root# efibootmgr -c -l \\EFI\\test\\abc.efi -L abc
```

Разумеется, загрузочные записи можно снова удалять. Для этого нужно указать номер записи с параметром -b:

```
root# efibootmgr -b 6 -Вц
```

Другие параметры команды efibootmgr описаны на странице справки man по этой команде.

15.4. GRUB 0.97

Во многих старых операционных системах, а также и в новых экземплярах некоторых дистрибутивов применяется загрузчик GRUB 0.97. В данном разделе я приведу важнейшую базовую информацию об этой версии GRUB, которая устарела уже несколько лет назад и официально не поддерживается.

Обратите внимание: GRUB 0.97 не может считывать ядро Linux ни из программного массива RAID, ни из LVM-тома. Кроме того, GRUB 0.97 несовместим со многими файловыми системами (например, с `btrfs`). Поэтому во многих случаях требуется создавать собственный загрузочный раздел, находящийся за пределами RAID и LVM и с такой файловой системой, которая поддерживает работу с GRUB 0.97 (например, ext2 или ext3). Во многих дистрибутивах присутствует пропатченная версия GRUB 0.97, совместимая как минимум с файловой системой ext4.

Конфигурация (файл меню)

Menu.lst. Организация меню GRUB определяется в файле меню GRUB, который обычно называется `/boot/grub/menu.lst`. По сравнению с конфигурацией GRUB 2, синтаксис этого файла достаточно прост. В отличие от ситуации с GRUB 2, здесь нет никаких сценариев, которые автоматически создавали или обновляли бы меню.

Особенности, специфичные для конкретных дистрибутивов. В большинстве дистрибутивов Linux файл меню GRUB обновляется при каждом обновлении ядра. Это позволяет гарантировать, что после перезапуска будет использоваться новейшая версия ядра. В любом случае не исключено, что при автоматическом обновлении конфигурации GRUB система удалит внесенные вами изменения.

В системах Debian и Ubuntu за обновление конфигурации GRUB отвечает сценарий `update-grub`. При создании новой версии конфигурационного файла сценарий работает с учетом специально обозначенных комментариев.

В Red Hat и Fedora командой, аналогичной `update-grub`, является `grubby`. Официальным конфигурационным файлом GRUB в RHEL, а также в старых версиях Fedora с GRUB 0.97 был файл `/etc/grub.conf`. Он представлял собой ссылку на действительный конфигурационный файл.

Для создания или конфигурирования GRUB в SUSE предназначен модуль YAST Система ▸ Конфигурация загрузчика.

Разделы. Обозначение разделов в GRUB 0.97 происходит примерно так же, как и в GRUB 2. Но есть два существенных отличия:

- нумерация разделов начинается с 0, а не с 1;
- перед номером раздела может стоять сокращение, указывающее на формат секционирования.

Итак, раздел, который в GRUB 2 назывался бы `(hd0,gpt3)`, в GRUB 0.97 именуется `(hd0,2)`.

В табл. 15.3 приведено несколько других примеров.

Таблица 15.3. Названия разделов GRUB

Название устройства GRUB 2	Значение
(hd0)	Первый жесткий диск (соответствует /dev/sda)
(hd1)	Второй жесткий диск (соответствует /dev/sdb)
(hd0,0)	Первый раздел первого жесткого диска (соответствует /dev/sda1)
(hd2,7)	Восьмой раздел третьего жесткого диска

Файл devices.map. На внутрисистемном уровне GRUB использует файл `/boot/grub/devices.map` для соотнесения приводов и названий устройств загрузчика. Этот файл создается при первом выполнении GRUB. Правда, он не обновляется автоматически при добавлении новых приводов. При необходимости `devices.map` можно просто удалить, а после этого выполнить GRUB. В таком случае система создаст файл заново. Этот процесс может занять около минуты.

В особо тяжелых случаях можете также попробовать самостоятельно изменить файл. Однако учитывайте, что вносимые изменения должны соответствовать информации, получаемой GRUB от BIOS при запуске компьютера. Формат файла таков:

```
# Пример /boot/grub/devices.map
(hd0) /dev/sda
(hd1) /dev/sdb
```

Глобальная область в menu.lst

Файл меню GRUB состоит из глобальной области, в которой содержатся различные базовые настройки, а также из нескольких записей меню, каждая из которых начинается со строки `title`. Например, в следующих строках показана глобальная область `menu.lst`:

```
# Глобальная область файла /boot/grub/menu.lst
default 2                    # Третья запись меню используется по умолчанию
timeout 30                   # Ожидать 30 секунд, пока не запустится
                             # система, заданная по умолчанию
color yellow/blue red/white  # Выделять записи меню цветом
```

В следующих абзацах описаны ключевые слова GRUB, используемые в глобальной области `menu.lst`.

- `default` — указывает номер записи меню, используемой по умолчанию. Счет начинается с 0! Вместо номера вы также можете задать `default saved`. В таком случае по умолчанию используется запись, применявшаяся при последнем запуске. Чтобы этот механизм работал, в каждой записи меню должно содержаться ключевое слово `savedefault` (подробнее об этом чуть ниже). Если в `menu.lst` отсутствует запись `default`, то первая запись в меню обозначает систему, запускаемую по умолчанию.
- `fallback` — задает номер записи меню, используемой, если запись, которая установлена для работы по умолчанию, содержит ошибку. Когда запись `fallback`

отсутствует, GRUB, сталкиваясь с подобными ошибками, переходит в интерактивный режим.

- `timeout` — указывает, в течение какого времени (в секундах) GRUB будет ожидать выбора из меню. По истечении этого времени автоматически запускается операционная система, указанная в качестве стандартной. Если вы хотите, чтобы GRUB ждала бесконечно, не запуская автоматически операционную систему, заданную по умолчанию, поставьте перед строкой `timeout` символ комментария #.
- `hiddenmenu` — заставляет GRUB не отображать меню. По истечении периода, указанного в `timeout`, запускается стандартная система. До наступления этого момента пользователь может отобразить меню, нажав Esc, а потом как обычно выбрать из него одну из записей.
- `password -- md5 code` — защищает GRUB паролем. Команды меню можно использовать и без пароля, но интерактивные функции GRUB предоставляются только после ввода пароля.
- `color fg/bg menufg/menubg` — управляет цветами, используемыми в меню GRUB. При этом команда `fg` задает цвет текста, а команда `bg` — цвет фона всего экрана. Соответственно, команды `menufg` и `menubg` определяют цвет выбранной записи меню. Если не использовать команду `color`, то меню GRUB отображается в черном и белом цветах.
- `splashimage` — имеется лишь в некоторых дистрибутивах Linux, в которых GRUB был соответствующим образом дополнен (например, в Fedora). Иначе говоря, `splashimage` не является официальной функцией GRUB. Это ключевое слово позволяет задавать для меню фоновый рисунок, который должен иметь размер 640 × 480 пикселов, быть в формате XPM (8 бит на пиксел) и находиться в архиве GZIP. В следующей строке показано, как использовать `splashimage`:

  ```
  splashimage=(hd1,1)/boot/grub/splash.xpm.gz
  ```

  Данный метод подробно описан в документе по адресу: http://www.owlriver.com/tips/hands-off/images.html.
- `gfxmenu` — еще одно неофициальное дополнение, предназначенное для отображения графических меню. Оно применяется, в частности, в дистрибутивах Novell и SUSE. Графический файл создается с помощью команды `mkbootmsg`. Эта команда, а также файл `gfxboot.html`, где содержится документация по ней, находится в пакете gfxboot. Так или иначе, чтобы самостоятельно создать сплэш-файл, вам придется потрудиться.

Записи меню в menu.lst

После глобальной области в `menu.lst` следуют записи меню, соответствующие различным операционным системам. Каждая запись меню вводится словом `title`. Текст, указанный после `title`, является содержанием строки меню. При этом испытания, которые я проводил, показали, что поддерживаются только символы US-ASCII, то есть международные специальные символы использовать нельзя. Мне не удалось найти никакой документации, где давалась бы подробная информация по кодировкам для `menu.lst`.

Остальные строки (вплоть до следующей команды `title` или до конца файла) — это команды GRUB, выполняемые в том порядке, в котором они записаны. (Если вы испытываете команды в интерактивном режиме, вам дополнительно потребуется выполнить `boot`. Эту команду нельзя указывать в файле меню.)

Запуск Linux. Чтобы запустить Linux, необходимо указать с помощью `root` раздел, в котором расположены ядро и файл диска оперативной памяти для начальной инициализации. Этот раздел будет считаться в GRUB активным. Команды `kernel` и `initrd` точно обозначают место, где находятся эти файлы, а также задают возможные параметры загрузки ядра.

Обратите внимание, что при определении параметров ядра (в частности, при указании корневого устройства ядра) используется номенклатура Linux. Корректная запись в данном случае будет выглядеть так: `root=/dev/sdb13`. Кроме того, есть и другие способы записи, например `root=LABEL=`*`label`* и `root=UUID=`*`n`*, причем в таких случаях необходимо указывать обозначение или идентификационный номер раздела.

В дальнейшем учитывайте, что названия устройств в вашей системе могут отличаться от `vmlinuz` и `initrd`.

```
# Выполнить запись меню в /boot/grub/menu.lst
# Linux в /dev/sdb13
title Linux
  root (hd1,12)
  kernel /boot/vmlinuz root=/dev/sdb13
  initrd /boot/initrd
```

Вы можете обойтись и без команды `root`. В таком случае потребуется указать для каждого файла нужный раздел диска:

```
# Выполнить Linux в /dev/sdb13
title Linux
  kernel (hd1,12)/boot/vmlinuz root=/dev/sdb13
  initrd (hd1,12)/boot/initrd
```

Если `/boot` находится не в системном разделе, а в собственном загрузочном разделе, нужно соответствующим образом изменить root-команду. Поскольку загрузочный раздел является исходной точкой для всех файлов, путь должен быть указан без `/boot`. В следующих строках подразумевается, что `/dev/sda2` является разделом `/boot` (решающее значение имеет строка `root (hd0,1)`).

```
# Запуск Linux в /dev/sdb13, если есть собственный загрузочный
# раздел /dev/sda2
title Linux
   root (hd0,1)
   kernel /vmlinuz root=/dev/sdb13
   initrd /initrd
```

UUID. В Ubuntu предусмотрена возможность указывать раздел с файлами ядра и `initrd` не только с помощью `root`, но и используя ключевое слово GRUB `uuid`. Правда, такой синтаксис действует лишь в Ubuntu, поскольку разработчики этого дистрибутива соответствующим образом модифицировали GRUB. Запись с применением `uuid` идеально подходит для установки Ubuntu на USB-носителях.

```
# Запуск Linux, вариант Ubuntu; загрузочный раздел имеет
# UUID=2a021cf3-..., корневой раздел управляется LVM
title Ubuntu
   uuid     2a021cf3-2b34-4cd4-b741-42b8fb1db89c
   kernel   /vmlinuz-3.1-13-generic root=/dev/mapper/vg1-ubuntu904 ro quiet
   initrd   /initrd.img-3.1-13-generic
```

Указание параметров ядра Linux. При запуске ядра сообщаются различные загрузочные параметры. Эти параметры, в частности, указывают, где находится системный раздел, и определяют, как отображаются сообщения системы Init-V. В файле `menu.lst` эти параметры просто записываются в конце строки `kernel`.

```
# Запуск Linux в /dev/sdb13 (с дополнительными параметрами ядра)
title Linux
   root (hd1,12)
   kernel /boot/vmlinuz root=/dev/sdb13 vga=normal
   initrd /boot/initrd
```

Запуск Windows. Если вы хотите запустить Windows, укажите активный раздел не с помощью `root`, а применив `rootnoverify`. Благодаря команде `chainloader +1` система прочитает и выполнит первый сектор этого раздела. В Windows 9x/ME это приводит к запуску операционной системы. В более новых версиях Windows, напротив, запускается загрузчик Windows, который, в свою очередь, уже загружает систему (сам GRUB не может запускать новейшие версии Windows).

```
# Запуск Windows в /dev/sda1
title Windows
   rootnoverify (hd0,0)
   chainloader +1
```

Запустить Windows удается лишь в том случае, если она находится на первом жестком диске. Если это не так, то следующие строки помогут вам виртуально поменять диски местами. Я не тестировал такой вариант.

```
# Запуск Windows в /dev/sdb1
title Windows
   rootnoverify (hd1,0)
   map (hd0) (hd1)
   map (hd1) (hd0)
   chainloader +1
```

Запуск другого загрузчика. Такие загрузчики, как GRUB, обычно устанавливаются в MBR первого жесткого диска. Однако имеется возможность устанавливать их в загрузочных секторах любых разделов. С помощью центрального GRUB, расположенного в MBR, можно опосредованно запустить другой GRUB, установленный в том или ином разделе. В `menu.lst` применяются те же ключевые слова, что и при запуске Windows:

```
# Запуск загрузчика в загрузочном секторе /dev/sda7
title Boot-Loader in /dev/sda7
   rootnoverify (hd0,6)
   chainloader +1
```

Такой метод может быть целесообразен, если требуется параллельно установить на одном жестком диске несколько дистрибутивов Linux.

Как пометить последнюю выбранную операционную систему. Если в `menu.lst` содержится много записей, стоит помечать запись, которую GRUB выбрал последней. Чтобы активизировать такую функцию, укажите в глобальной области `menu.lst` параметр `default saved` и дополните все записи меню ключевым словом `savedefault`:

```
# Запуск Linux в /dev/sdb13; пометить сделанный выбор
title Linux
   root (hd1,12)
   kernel /boot/vmlinuz root=/dev/sdb13
   initrd /boot/initrd
   savedefault
```

ВНИМАНИЕ

Ни в коем случае не применяйте savedefault на компьютерах, где в BIOS расположено два или более жестких диска, объединенных в группу RAID (BIOS-Software-RAID)! Вы рискуете потерять данные либо нарушить синхронность работы системы!

Тестирование конфигурации GRUB

Если хотите быстро и без перезапуска проверить, нет ли в измененном файле меню GRUB синтаксических ошибок, запустите GRUB и выполните в нем следующую команду:

```
root# grub
grub> configfile (hd1,12)/boot/grub/menu.lst
```

Вместо `(hd1,12)` следует указать название GRUB для того раздела жесткого диска, где находится файл меню загрузчика. Если все получится, то GRUB отобразит меню. Правда, вы не сможете запустить ни одну из операционных систем, поскольку у вас уже работает Linux.

Сценарий update-grub (Debian и Ubuntu)

В Debian до версии 5 и Ubuntu до версии 9.04 для обновления конфигурации GRUB предусмотрен сценарий оболочки `update-grub`. Он ищет в каталоге `/boot` файлы `vmlinuz-*` и добавляет для каждого такого файла ядра отдельную запись в меню. При этом также учитываются файлы `initrd-*`, соответствующие той или иной версии ядра. Сценарий `update-grub` автоматически выполняется после каждого обновления ядра.

Сценарий учитывает некоторые настройки, которые указываются в `menu.lst`, как комментарии. Отдельные переменные подробно описаны в файле справки `man update-grub`. Самая важная переменная называется `kopt` и указывает, какие параметры сообщаются ядру. Если вы хотите внести в `menu.lst` изменения, касающиеся этой переменной, не изменяйте записи в `menu.lst` напрямую — изменяйте саму переменную `kopt`. Затем еще раз выполните `update-grub`.

```
# /etc/boot/menu.lst
...
# Настройки для update-grub
#
# kopt=root=/dev/sda13 ro  (Параметры ядра)
# groot=(hd0,12)           (Место, в котором установлен GRUB)
# alternative=true         (Добавление записей о других операционных системах)
# lockalternative=false    (Опустить записи о других операционных системах)
# defoptions=              (Параметры, которые касаются лишь ядра, используемого
                           по умолчанию)
# lockold=false            (Опустить параметры, касающиеся других операционных
систем)
# xenhopt=                 (Параметры Xen)
# xenkopt=console=tty0     (Параметры ядра для ядра Xen)
# altoptions=(single-user mode) single (Параметры ядра для альтернативного ядра)
# howmany=all              (Максимальное количество записей, касающихся ядра)
# memtest86=true           (Добавить запись Memtest)
# updatedefaultentry=false (Изменить ключевое слово, используемое в menu.lst по
                            умолчанию)
...
```

Grubby (Fedora, Red Hat)

В RHEL и Fedora за обновление конфигурации GRUB после обновления ядра отвечает сценарий `grubby`. Он хорошо документирован на соответствующей странице в справке `man`.

Но там же специально указано, что `grubby` следует использовать исключительно для внутрисистемного вызова сценариев обновления, специфичных для Red Hat, но нельзя применять при работе вручную.

GRUB 0.97 и EFI

В Red Hat и Fedora GRUB 0.97 применяется и при установках с EFI. Но официально GRUB 0.97, конечно же, не поддерживает EFI. Fedora и Red Hat просто используют пропатченную версию загрузчика. По сравнению с невероятно сложным GRUB 2 вся прелесть GRUB 0.97 для работы с EFI заключается именно в простоте.

Загрузчик находится в неизменяемом файле `grub.efi`, который устанавливается и на компьютерах с BIOS в каталоге `/boot/efi/EFI/redhat`. Этот файл входит в состав пакета `grub`. Находится ли `/boot/efi` в собственном разделе, не проверяется. Если позже вы решите переделать имеющуюся установку с BIOS в установку с EFI, вы должны обязательно убедиться, что раздел EFI подключается к системе именно в точке `/boot/efi`.

`grub.efi` ожидает найти в этом же каталоге конфигурационный файл. Поэтому при установке с EFI `/etc/grub.conf` представляет собой ссылку на `/boot/efi/EFI/redhat/grub.conf`. Синтаксис `grub.conf` такой же, как и в BIOS-версии GRUB 0.97. При установках с EFI в этом же каталоге сохраняется и файл `devices.map`.

Исправление установки GRUB с помощью «живого диска»

После запуска «живого диска» откройте консоль и выполните в ней команду `su -l` или `sudo -s`, чтобы получить права администратора. Далее нужно найти раздел, в котором находится ваш дистрибутив Linux и, в частности, каталог `/boot`.

```
root# mkdir /test
root# mount /dev/sda3 /test
```

Искомый раздел вы узнаете по наличию каталога `/boot` с файлами ядра Linux (`vmlinuzxxx`), подкаталогу `/boot/grub` и файлу меню GRUB `/boot/grub/menu.lst`. Если на компьютере создан специальный загрузочный раздел, то в этом разделе не будет каталога `boot`. `vmlinuzxxx` и подкаталог `grub` в таком случае будут находиться прямо в корневом каталоге загрузочного раздела.

После завершения подготовительных работ нужно позаботиться о том, чтобы установить GRUB в загрузочный сектор жесткого диска. Тогда при запуске компьютера будут считаны все остальные файлы GRUB, находящиеся в `/dev/sda3`. При этом раздел `/dev/sda3` обычно обозначается в GRUB как `(hd0,2)`, а `(hd0)` служит обозначением всего первого жесткого диска, в загрузочный сектор которого должен быть записан GRUB.

```
root# grub
grub> root (hd0,2) (Здесь находится системный, то есть загрузочный раздел)
grub> setup (hd0)  (Место назначения установки GRUB: MBR первого жесткого диска)
grub> quit
```

16 Система Init

В этой главе описаны процессы, протекающие в период от запуска ядра до запроса данных для входа в систему. После запуска ядро временно может обратиться к корневому разделу лишь в режиме «только для чтения». В первую очередь ядро запускает программу `/sbin/init`.

В дальнейшем программа `init` отвечает за базовую конфигурацию системы (подключение файловых систем) и запуск многочисленных сетевых служб и демонов.

Как это часто случается в мире Linux, одновременно существует не один, а несколько вариантов системы Init. В настоящее время наиболее распространены классическая система Init-V, Upstart и Systemd.

- В классической системе Init-V за инициализацию компьютеров отвечает множество сценариев, связанных между собой ссылками. Сама концепция и название системы восходят к системе V, применявшейся в операционной системе UNIX. В некоторых дистрибутивах система Init-V используется по умолчанию, но в данной книге описан только один такой дистрибутив — Debian. В более ранних дистрибутивах система Init-V встречается чаще. Но все современные дистрибутивы по-прежнему совместимы с Init-V. Поэтому вам не помешает на базовом уровне ознакомиться с этой системой.
- Upstart — это событийно-ориентированный вариант системы Init. В Ubuntu он применяется в версии 6 и выше, а также присутствует в Fedora в версиях с 9 по 13 и в RHEL 6.
- Systemd — самая современная из существующих Init-систем, которая впервые стала применяться в Fedora 15. openSUSE также перешел на эту систему, начиная с версии 12.1.

В табл. 16.1 обобщается информация о том, какая система Init применяется в каком дистрибутиве. Столбец RHEL относится к Red Hat Enterprise Linux, а также ко всем совместимым с ним дистрибутивам, например CentOS.

Таблица 16.1. Init-системы с указанием дистрибутивов, в которых они применяются

Система Init	Debian	Fedora	openSUSE	RHEL	Ubuntu
Init-V	До 7	До 8	До 11.4	До 5	До 6.04
Upstart	—	С 9 по 13	—	6	С 6.10
Systemd	—	С 14	С 12.1	С 7	—

В этой главе в общем виде рассмотрены сначала Init-V, Upstart и Systemd, а потом обсуждаются детали запуска системы в современных версиях Devian, Fedora, openSUSE и Ubuntu.

Заканчивается глава рассказом о Internet Service Daemons — программе, которая отслеживает работу сетевых портов. При поступлении запросов с определенного порта Internet Service Daemons запускает программу, способную обработать такой запрос.

16.1. Init-V

В этом разделе описываются концепции, лежащие в основе классической системы Init-V. Она применяется в некоторых современных и многих более ранних дистрибутивах для того, чтобы инициализировать систему и запускать сетевые службы.

От конкретного дистрибутива зависит, как именно будет выполняться программа `/sbin/init`: какие файлы `init` в каких каталогах расположены, какими номерами и буквами обозначаются так называемые уровни запуска, какие конфигурационные файлы учитываются при запуске и т. д.

Хотя в современных версиях Fedora, openSUSE и Ubuntu для запуска системы применяются Upstart и Systemd, в данном разделе мы также не раз коснемся этих дистрибутивов. Дело в том, что и Upstart, и Systemd совместимы с Init-V, а некоторые принципы, действующие в Init-V, применяются и в новых дистрибутивах.

Процесс. Чтобы при чтении не запутаться в многочисленных деталях, рассмотрим сначала обычный запуск Linux с применением системы Init-V.

1. GRUB загружает и запускает ядро.
2. Ядро запускает программу `/sbin/init`.
3. Программа `init` интерпретирует конфигурационный файл `/etc/inittab`.
4. Программа `init` выполняет сценарий для инициализации системы.
5. Программа `init` выполняет сценарий `/etc/rc.d/rc` или `/etc/init.d/rc`. Сценарий `rc` во многом различен в разных дистрибутивах. Он нужен для запуска файлов сценариев, находящихся в каталоге `/etc/rc`*n*`.d` или `/etc/init.d/rc`*n*`.d` (*n* в данном случае — уровень запуска, см. ниже). Кроме того, `rc` активизирует в большинстве дистрибутивов и графический индикатор загрузки, который показывает, насколько выполнен процесс Init-V.
6. Файлы сценариев из `/etc/rc`*n*`.d` или `/etc/init.d/rc`*n*`.d` запускают различные системные службы, в частности те, которые отвечают за выполнение сетевых функций.

Уровень запуска

Сначала ядро запускает программу `/sbin/init`. При этом сообщаются все параметры загрузки ядра, которые еще не были интерпретированы (то есть все те параметры,

которые ядро не знает и, соответственно, не может самостоятельно обработать). В результате этого Linux можно, например, запустить в однопользовательском режиме (подробности сообщаются на странице `man init`). Иными словами, `init` — это первый работающий процесс. Все остальные процессы запускаются либо процессом `init` непосредственно, либо его подпроцессами (если вы выполните `pstree`, то сразу поймете, какую доминирующую роль играет процесс `init`). При остановке компьютера процесс `init` последним завершает работу, обеспечивая правильное завершение всех остальных процессов.

Уровни запуска в Fedora, Red Hat, SUSE. Для понимания того, как работает System-V, нужно прежде всего усвоить, что такое *уровень запуска*. Уровни запуска описывают различные состояния, в которых может находиться операционная система. К сожалению, нумерация уровней запуска в различных дистрибутивах неодинакова. Как правило, значения уровней запуска в конкретном дистрибутиве документированы в `/etc/inittab`.

В большинстве дистрибутивов (но не в Debian и Ubuntu!) значения уровней запуска таковы:

- `0` — остановка работы компьютера с выключением;
- `1` и `S` — однопользовательский режим;
- `2` — многопользовательский режим без выхода в сеть и без NFS;
- `3` — многопользовательский режим с выходом в сеть, но без автоматического запуска X;
- `4` — обычно не применяется;
- `5` — многопользовательский режим с выходом в сеть и запуском системы X; установлен по умолчанию;
- `6` — остановка работы компьютера с перезагрузкой.

В некоторых системах существует различие между уровнями запуска `1` и `S`. При работе с такими системами выполняются специальные сценарии уровня запуска `1`, позволяющие перейти с обычного уровня (`2`, `3` или `5`) на уровень запуска однопользовательского режима. Напротив, сценарии уровня запуска `S` применяются лишь тогда, когда вам требуется активировать уровень запуска однопользовательского режима сразу после загрузки (параметр ядра `single`).

Уровни запуска в Debian и Ubuntu. В дистрибутивах, построенных на основе Debian, уровни запуска `2-5` равноправны. На каждом их этих уровней запускается многопользовательская система с выходом в сеть и X. Стандартным уровнем запуска считается `2`. Уровень запуска `S` не является уровнем в полном смысле этого слова, он просто применяется для инициализации компьютера сразу после запуска (еще до того, как будет активизирован какой-либо иной уровень). Сетевые функции активизируются в Debian и Ubuntu уже на этапе инициализации системы (то есть перед включением того или иного уровня запуска), следовательно, они доступны на всех уровнях запуска:

- `S` — инициализация компьютера непосредственно после запуска;
- `0` — остановка работы компьютера с выключением;
- `1` — однопользовательский режим с доступом к сети;

- 2-5 — многопользовательский режим с доступом к сети и автоматическим запуском X;
- 6 — остановка работы компьютера с перезагрузкой.

Смена уровня запуска. Администратор может изменить уровень запуска с помощью команды `init x`, не останавливая работу системы. Здесь *x* — это цифра или буква, означающая уровень запуска. Например, бывает полезно изменить уровень запуска при проведении некоторых работ по техническому обслуживанию компьютера. При выполнении команды `shutdown` или нажатии Ctrl+Alt+Delete уровень запуска также изменяется, в результате чего происходит перезапуск компьютера.

Стандартный уровень запуска. В классической системе Init-V стандартный уровень запуска определяется строкой `initdefault` в `/etc/inittab`. В большинстве современных дистрибутивов стандартным уровнем запуска является 5; в Debian и Ubuntu — 2.

В Ubuntu версии 9.10 и выше стандартный уровень запуска определяется в `/etc/init/rc-sysinit.conf`. Эти файлы водят в состав системы Upstart.

В дистрибутивах, использующих Systemd, стандартный уровень запуска определяется в файле `/etc/systemd/default.target`. Эта ссылка указывает на заранее определенные целевые файлы в каталоге `/lib/systemd/system/`.

Inittab

При запуске системы команда `init` управляется файлом `/etc/inittab`. Синтаксис записей осуществляется по следующей схеме:

```
id-code:runlevel:action:command
```

`id-code` состоит из двух символов, однозначно идентифицирующих строку. `runlevel` указывает, на каком уровне запуска выполняется запись. `action` содержит команду для `init`. `command` указывает, какую программу или команду Linux следует запустить. В табл. 16.2 перечислены важнейшие ключевые слова для `action` (полное описание дается в `man inittab`).

Таблица 16.2. Ключевые слова

Ключевое слово	Значение
ctrlaltdel	Указывает, как init должна реагировать на Ctrl+Alt+Delete
initdefault	Определяет стандартный уровень запуска для init (см. выше)
once	init выполняет указанную команду при смене уровня запуска
respawn	После окончания выполнения команды init снова ее запускает
sysinit	init один раз выполняет команду в процессе загрузки
wait	init ожидает окончания выполнения следующей команды
bootwait	init запускает команду в процессе загрузки и ожидает окончания выполнения следующей команды

В следующем листинге показан немного сокращенный файл `inittab` из Debian 6. Стандартный уровень запуска — 2. При обычном запуске системы `init` выполняет сценарные файлы `rcS` и команду `rc 5`. Наконец, в текстовых консолях 1-6 запускается программа `mingetty`, обеспечивающая вход в систему. (Если вам нужно больше

текстовых консолей, изменить их количество следует именно здесь. Однако обратите внимание на то, что в большинстве дистрибутивов консоль 7 зарезервирована для X.)

```
# Файл /etc/inittab в Debian 6
# Стандартный уровень запуска
id:2:initdefault:

# Конфигурация и инициализация системы производится сразу после запуска
# компьютера
si::sysinit:/etc/init.d/rcS

# Работа в однопользовательском режиме (параметр ядра su)
~~:S:wait:/sbin/sulogin

# Старт соответствующего уровня запуска
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
# Следующую строку не нужно использовать,
# она предусмотрена для аварийных ситуаций
z6:6:respawn:/sbin/sulogin

# Реакция на Ctrl+Alt+Delete в текстовой консоли
ca:12345:ctrlaltdel:/sbin/shutdown -t1 -a -r now

# Реакция на перебой с электропитанием
# (необходимо наличие источника бесперебойного питания)
pf::powerwait:/etc/init.d/powerfail start
pn::powerfailnow:/etc/init.d/powerfail now
po::powerokwait:/etc/init.d/powerfail stop

# Запуск gettys (эмуляторов терминалов) для текстовых консолей
1:2345:respawn:/sbin/getty 38400 tty1
2:23:respawn:/sbin/getty 38400 tty2
3:23:respawn:/sbin/getty 38400 tty3
4:23:respawn:/sbin/getty 38400 tty4
5:23:respawn:/sbin/getty 38400 tty5
6:23:respawn:/sbin/getty 38400 tty6
```

Изменение действия сочетания Ctrl+Alt+Delete. Большинство дистрибутивов по умолчанию сконфигурированы так, что сочетание клавиш Ctrl+Alt+Delete в текстовом режиме приводит к перезапуску компьютера. Если вы хотите, чтобы вместо этого компьютер выключался, укажите в команде `shutdown` в строке `ca:` параметр `-h` вместо `-r`. Если хотите вообще отключить это сочетание клавиш, поставьте перед строкой `ca:` символ комментария #. Но учитывайте, что изменять файл `inittab` целесообразно лишь в тех дистрибутивах, которые используют систему Init-V.

Инициализация системы

Еще до того как запустятся либо завершат работу описанные далее файлы rc и runlevel-специфичные службы, то есть сразу после запуска компьютера, выполняется инициализация системы (строка si: в inittab). Название сценария зависит от дистрибутива:

- Debian, Ubuntu до версии 6.06 — /etc/init.d/rcS;
- RHEL 5, Fedora до версии 8 — /etc/rc.d/rc.sysinit;
- openSUSE до версии 11.4 — /etc/init.d/boot.

В ходе инициализации системы решаются задачи, которые требуется выполнить только один раз, в начале сеанса работы с компьютером:

- инициализация различных системных переменных (в том числе хост-имен и доменных имен);
- активизация файловой системы /proc;
- настройка даты и времени;
- настройка раскладки клавиатуры для текстовой консоли;
- запуск системы udev;
- активизация RAID или LVM;
- перепроверка файловых систем;
- подключение корневого раздела заново, в режиме «для чтения и внесения изменений»;
- проверка файловой системы следующего раздела, подключение разделов;
- частичная (Fedora) или полная (Debian, Ubuntu) инициализация основных сетевых функций.

Обратите внимание — не все описанные здесь функции напрямую выполняются сценарием инициализации системы. Частично считываются и другие сценарные файлы. При этом их названия обычно записываются в форме . *имя* (поставив точку, мы гарантируем, что находящийся в этом месте файл будет прочтен и выполнен; затем продолжится выполнение сценария). В Debian и Ubuntu присутствует миниатюрный сценарий rcS, выполняющий все сценарные файлы /etc/rcS.d/S*.

Сценарии Init-V для активации уровней запуска

После инициализации системы активизируется стандартный уровень запуска в соответствии с /etc/inittab. Для всех действий, связанных с уровнем запуска, существуют собственные сценарные файлы. В зависимости от дистрибутива они могут находиться в каталоге /etc/init.d или /etc/rc.d/init.d. Чтобы достичь большей совместимости между дистрибутивами, лучше указывать оба пути в виде ссылок.

Для запуска сценария Init-V программа init выполняет сценарий /etc/rc.d/rc или /etc/init.d/rc. Сценарию rc передается желаемый уровень запуска *n*. Сначала rc выполняет некоторые операции, нужные для инициализации. После этого

выполняются все сценарные файлы rc*n*.d/K*, необходимые для завершения процессов. Наконец, выполняются все файлы rc*n*.d/S*, требуемые для активации новых процессов на том или ином уровне запуска.

Значительное достоинство этой малопонятной системы заключается в том, что вы можете с легкостью интегрировать новые системные процессы в процесс Init-V. Нужно просто скопировать сценарии rc для запуска и остановки процессов в правильные каталоги — именно это происходит при установке пакета, содержащего дополнительный демон. Следующий вывод для Debian 5 демонстрирует, какие сценарные файлы выполняются для некоторых уровней запуска.

```
user$  cd /etc/rc.d
user$ ls rcS.d/ rc2.d/ rc6.d/
rcS.d/: (Инициализация системы)
S01mountkernfs.sh  S07checkroot.sh       S12mountoverflowtmp S16mountnfs.sh
S02udev            S08hwclock.sh         S13networking   S17mountnfs-boo...
S03mountdevsubfs.sh   S08ifupdown-clean  S13pppd-dns     S18console-scre...
S04bootlogd        S08module-init-tools  S13procps       S19console-setup
S05keyboard-setup  S08mtab.sh            S13udev-mtab S  20alsa-utils
S06hdparm          S09checkfs.sh         S13urandom      S20bootmisc.sh
S06hostname.sh     S10ifupdown           S13x11-common   S20fuse
S06hwclockfirst.sh S10mountall.sh        S14portmap      S21stop-bootlog...
S06lvm2            S11mountall-bootclean.sh S15nfs-common

rc2.d/:rc2.d/: (Стандартный уровень запуска)
S01vboxadd         S18acpid           S18ssh             S21bootlogs
S02vboxadd-service S18anacron         S19avahi-daemon    S22rc.local
S14portmap         S18atd             S19bluetooth       S22rmnologin
S15nfs-common      S18cron            S19cpufrequtils    S22stop-bootlogd
S17binfmt-support  S18dbus            S19network-manager
S17rsyslog         S18exim4           S20gdm3
S17sudo            S18loadcpufreq     S20saned

c6.d/: (Завершение работы)
K01alsa-utils      K01network-manager    K03sendsigs     K08ifupdown
K01anacron         K01saned              K04rsyslog      K09umountfs
K01atd             K01unattended-upgradesK05umountnfs.sh K10lvm2
K01bluetooth       K01urandom            K06nfs-common   K11umountroot
K01exim4           K01vboxadd-service    K06portmap      K12reboot
K01fuse            K02avahi-daemon       K07hwclock.sh
K01gdm3            K02vboxadd            K07networking
```

Если быть точным, в каталогах rc*n*.d находятся не сами сценарные файлы, а только ссылки на них. Преимущество этого метода заключается в том, что каждый сценарный файл можно использовать для нескольких уровней запуска и централизованно изменять. Сами сценарные файлы сохраняются в каталоге /etc/rc.d/init.d или /etc/init.d:

```
root# cd /etc
root# ls -l rc2.d/S20cups
... rc2.d/S20cups -> ../init.d/cups
```

Система именования

Названия ссылок не произвольны, хотя на первый взгляд может сложиться такое впечатление. Начальная буква указывает, запускает сценарий процесс (`S=start`) или завершает его (`K=kill`). Ссылки `S` и `K` указывают на одни и те же файлы, но в зависимости от начальной буквы `rc` выполняет сценарий или с параметром `start`, или с параметром `stop`.

Указанный далее номер определяет очередность, в которой выполняются сценарные файлы. Например, для запуска большинства сетевых демонов требуется, чтобы сетевое соединение уже было налажено, после чего они запускаются сценарием `network`.

Как автоматически запускать демоны и завершать их работу

Многие сценарные файлы уровней запуска автоматически выполняются сценарием `rc` при включении компьютера или при изменении уровня запуска. При этом сообщается параметр `start` или `stop` в зависимости от того, следует ли запустить или остановить соответствующую функцию.

Чтобы можно было выполнить автоматический запуск или остановку, требуется ссылка `S` или `K` на сценарий Init-V для соответствующего уровня запуска в каталоге `rcn.d`. Иными словами, если вы хотите, чтобы в дальнейшем определенная функция активизировалась автоматически, вам нужно создать такие ссылки. Соответственно, чтобы отменить автоматический запуск, следует удалить такие ссылки.

Следующие команды показывают, какие ссылки необходимо создать в Red Hat, чтобы в дальнейшем программа Samba автоматически начинала работу на уровнях запуска `2` и `5`:

```
root# cd /etc/
root# ln init.d/samba rc0.d/K01samba
root# ln init.d/samba rc1.d/K01samba
root# ln init.d/samba rc2.d/S17samba
root# ln init.d/samba rc3.d/S17samba
root# ln init.d/samba rc4.d/S17samba
root# ln init.d/samba rc5.d/S17samba
root# ln init.d/samba rc6.d/K01samba
```

Удаление ссылок сильно упрощается:

```
root# rm rc?.d/*samba
```

На практике вам достаточно редко придется вводить вручную вышеуказанные команды `ln` или `rm`. В большинстве дистрибутивов имеются специальные команды, избавляющие вас от этой работы, например `insserv` в Debian 6 и выше, а также в SUSE, `chkconfig` в RHEL5 и старых версиях Fedora и Mandriva или `update-rc.d` в старых версиях Debian и Ubuntu.

Как вручную запускать и останавливать демоны

Сценарные файлы уровней запуска можно выполнять и вручную. Например, следующая команда останавливает работу веб-сервера Apache:

```
root# /etc/init.d/samba stop
```

Во многих дистрибутивах также предусмотрены специальные команды для запуска и остановки работы демонов, избавляющие вас от излишней работы с клавиатурой. Наиболее распространенной из таких команд является `service`. Она имеется во всех дистрибутивах, описанных в книге, независимо от применяемой разновидности системы Init-V (но в Debian — только в версии 7 и выше!).

```
root# service samba start
```

Параметры Init-V-Script

В большинстве сценариев могут быть переданы следующие параметры:

- `start` — запускает указанную функцию;
- `stop` — останавливает функцию;
- `status` — показывает информацию, активна функция или нет;
- `reload` — используется, когда измененные конфигурационные файлы нужно заново прочитать без полной остановки демона при этом;
- `restart` — по сравнению с предыдущим параметром полностью останавливает демона и запускает его заново. Установленные связи с клиентом при этом теряются.

Во многих демонах параметры `reload` и/или `restart` не предусмотрены. В этом случае нужно завершать скрипт с помощью `stop` и затем запускать, используя `start`.

chkconfig (Fedora, Red Hat)

В Fedora и RHEL команда `chkconfig` обеспечивает управление ссылками на сценарии уровней запуска. При применении параметра `--list` эта команда выдает обзор всех сценариев и указывает, на каком уровне они запускаются. Если у вас установлена `xinetd`, то перечисляются и ее службы.

```
root# chkconfig --list
NetworkManager  0:Off 1:Off 2:On  3:On  4:On     5:On     6:Off
NetworkManagerD 0:Off 1:Off 2:Off 3:Off 4:Off    5:Off    6:Off
acpid           0:Off 1:Off 2:Off 3:On  4:On     5:On     6:Off
...
Службы на основе xinetd:
      chargen-udp: Off
      ...
```

Параметр `--del` позволяет в принципе отменить запуск сценариев уровня запуска. Сам сценарий сохраняется, удаляются только ссылки в каталогах `rc`*n*`.d`.

```
root# chkconfig --del samba
```

Параметр `chkconfig --add` добавляет во всех предусмотренных уровнях запуска ссылки для запуска и остановки, относящиеся к новой службе. Правда, параметр `--add` функционирует лишь при условии, что файл сценария Init-V явно содержит информацию о том, на каком уровне запуска сценарий должен по умолчанию запускаться.

По многим сценариям такая информация отсутствует. Чтобы в будущем подобный сценарий запускался автоматически, нужно использовать `chkconfig --level n`

name on/off. Такой вариант синтаксиса удобен и в тех случаях, когда вы, отступая от стандартных значений, хотите сами задать, на каком уровне запуска должен работать конкретный сценарий.

Например, в сценарном файле /etc/rc.d/init.d/httpd не содержится никакой информации о том, на каких уровнях следует автоматически запускать сервер. Поэтому chkconfig --add создает лишь ссылки для завершения работы веб-сервера. Chkconfig --level добавляет ссылки для запуска веб-сервера на уровнях 3 и 5. chkconfig --list выводит результат:

```
root# chkconfig --add httpd
root# chkconfig --level 35 httpd on
root# chkconfig --list httpd
httpd 0:Off 1:Off 2:Off 3:On  4:Off 5: On 6: Off
```

Команды chkconfig --add и chkconfig --del работают и со службами xinetd. Xinetd — это демон, задача которого заключается в том, чтобы запускать службы лишь по необходимости. В качестве альтернативы chkconfig для администрирования Init-V предлагается графический пользовательский интерфейс system-config-services. Эту программу можно запустить и с помощью команды serviceconf. Она позволяет вручную запускать и останавливать, а также перезапускать отдельные демоны и службы xinetd. Кроме того, можно (для заранее выбранного уровня запуска) щелчком кнопки мыши указать, какие службы должны запускаться по умолчанию.

Оптимизация процесса Init-V

Вы, вероятно, уже поняли, что процесс Init-V достаточно сложен: в ходе этого процесса требуется запускать и останавливать многочисленные сценарии и программы. Для каждого отдельного сценария должен выполняться специальный интерпретатор (как правило, bash или dash). Еще несколько лет назад запуск системы мог занимать одну-две минуты. Однако в настоящее время в большинстве дистрибутивов с помощью разнообразных приемов удалось снизить это время примерно до полуминуты. В этом подразделе обобщены самые распространенные методы оптимизации загрузки системы.

- **Загрузка файлов «заранее».** При каждом запуске с жесткого диска считываются одни и те же файлы в одинаковом порядке. При считывании файлов, конечно же, возникают небольшие временные промежутки (периоды простоя системы). Чтобы ускорить такой процесс считывания, некоторые дистрибутивы в начале выполнения процесса Init-V запускают фоновую программу, считывающую все файлы, которые указаны в специальном списке. Если какой-либо файл действительно понадобится в ближайшее время, он сразу помещается в кэш.

  Независимо от системы Init-V эта же идея применяется в Upstart и Systemd. В RHEL за эти задачи отвечает программа readahead, в Ubuntu — ureadahead. В Fedora и openSUSE в конфигурации Systemd содержатся службы readahead. Только Debian отказывается от оптимизации такого рода. Учитывая все более

широкое распространение твердотельных дисков, это, вероятно, самое перспективное решение.

- **Распараллеливание.** Обычно процесс Init-V протекает последовательно, то есть сценарии Init-V обрабатываются один за другим. Так происходит потому, что невозможно подключить к файловой системе каталог NFS, пока не инициализированы сетевые функции. Кроме того, практически невозможно обратиться к файлам каталога NFS до того, как этот каталог будет интегрирован в файловую систему. Но не все сценарии Init-V зависимы друг от друга. Так, например, вы вполне можете параллельно запустить функции настройки сети и инициализацию печати, почтового сервера и веб-сервера. Поэтому в некоторых дистрибутивах предпринимаются попытки параллельно выполнять те сценарии Init-V, которые не зависят друг от друга.
- **Более ранний запуск графической системы X.** Раньше было принято запускать систему X ближе к концу процесса Init-V. Но в наши дни система X запускается еще до того, как завершится основная инициализация оборудования. Ранний запуск X является в определенной степени частным случаем распараллеливания процесса Init-V.

16.2. Upstart

Ubuntu был первым крупным дистрибутивом, сделавшим шаг в сторону от системы Init-V и приступившим в версии 6.10 к использованию Upstart. Fedora последовала за Ubuntu в своей версии 9, но вновь отказалась от этой системы в версии 15, перейдя уже к работе с Systemd. RHEL 6 также применяет Upstart. Но начиная с версии RHEL 7, этот дистрибутив также пополнит ряды приверженцев Systemd. В долгосрочной перспективе Ubuntu должна остаться единственным крупным дистрибутивом, применяющим Upstart.

Далее будут рассмотрены важнейшие концепции и конфигурационные файлы версии Upstart 1.3 на базе Ubuntu 12.10. Обратите внимание, что в более ранних версиях Upstart вместо части рассмотренных ниже каталогов конфигурации используются другие каталоги. Более подробная информация сообщается в `init`, `initctl`, `telinit` и `runlevel` и на следующих сайтах:

- http://upstart.ubuntu.com/;
- http://upstart.ubuntu.com/cookbook/;
- http://docs.redhat.com/docs/en-US/Red_Hat_Enterprise_Linux/6/html/Technical_Notes/deployment.html.

Концепция. Концепция Upstart принципиально отличается от Init-V. Upstart — это событийно-ориентированная программа. События происходят при запуске и остановке программ или служб. Upstart обрабатывает события и реагирует на них, запуская или останавливая (другие) службы или инициируя другие события. События также обеспечивают простой обмен информацией между двумя процессами.

Как и в случае с Init-V, программа `/sbin/init` запускается ядром как первый процесс. Однако программа `init` входит в состав Upstart и не имеет ничего общего

с программой init системы Init-V! Описываемая здесь init в основном занимается интерпретацией конфигурационных файлов и реагирует на события.

Последовательность событий инициируется командой startup. Это событие автоматически происходит после запуска /sbin/init. Ради совместимости с Init-V в Upstart также предусмотрены уровни запуска. Они реализованы в новой системе как события с названиями runlevel *n*.

Конфигурация. Файла /etc/inittab в данном случае нет. Вместо этого все конфигурационные файлы находятся в каталоге /etc/init. Программа /sbin/init считывает все CONF-файлы из этого каталога. Типичный конфигурационный файл выглядит так:

```
# /etc/init/tty1.conf
start on stopped rc RUNLEVEL=[2345] and (
      not-container or
      container CONTAINER=lxc or
      container CONTAINER=lxc-libvirt)
stop on runlevel [!2345]
respawn
exec /sbin/getty -8 38400 tty1
```

Данный файл отвечает за запуск эмулятора терминала (программы getty) в текстовой консоли 1. Команда exec выполняется не сразу, а только тогда, когда произойдет одно из событий, определенных start, то есть после завершения работы сценариев rc2-rc5. Аналогичным образом работа getty завершается, когда произойдет одно из событий, указанных в stop on. Благодаря respawn программа getty автоматически перезапускается, если ее работа прекратится незапланированным образом.

Кроме ключевых слов, указанных выше, есть еще некоторые: команду о запуске программы можно сформулировать не только с помощью exec, но и посредством script / end script. Все строки, находящиеся между этими командами, будут выполнены оболочкой /bin/sh. Если при запуске или остановке работы потребуются дополнительные мероприятия по инициализации или настройке, то соответствующие команды будут сформулированы с помощью per-start script / end script или post-stop script / end script. Благодаря указанию console output результаты выполнения программ выводятся на консоль, а не переадресовываются по умолчанию на /dev/null (вывод игнорируется). За реагирование на нажатие клавиш Ctrl+Alt+Delete отвечает файл control-alt-delete.conf.

Если самому конфигурационному файлу Upstart требуется инициировать какое-либо событие, то применяется команда emits eventname. По некоторым событиям в справке man даже есть отдельные страницы (см., например, man local-filesystems). Обратите внимание на то, что некоторые события Upstart создаются сценариями, находящимися вне /etc/init! Например, событие ifup сценария /etc/network/if-up.d/upstart инициируется командой initctl. События, связанные с уровнями запуска, выполняются командой telinit.

Совместимость с Init-V. Upstart в том виде, в котором он существует сейчас, запускает далеко не все системные службы. Чтобы запустить оставшиеся службы, используется один из сценариев, обеспечивающих совместимость с Init-V, — /etc/init.d/rc или /etc/rc.d/rc. За его запуск отвечают конфигурационные файлы Upstart

rc.conf и rcS.conf. Сценарий /etc/init.d/rc работает в полном соответствии требованиям системы Init-V, то есть все сценарии, запускающие и останавливающие программы, выполняются в /etc/rcn.d.

Однако предусмотрена и обратная совместимость: чтобы службы, управляемые Upstart, можно было запускать командами системы Init-V, каталог /etc/init.d/ служб Upstart содержит ссылку на сценарий /lib/init/upstart-job. Этот сценарий отвечает за выполнение подходящих команд Upstart. Итак, если вы выполняете в Ubuntu 9.10 /etc/init.d/gdm, сценарий upstart-job выполняет команду stop gdm. Взглянув на ссылки в /etc/init.d, вы сразу же поймете, сколько служб уже выполняется с помощью Upstart (ls -l /etc/init.d).

В настольных версиях Ubuntu совместимость с Init-V уже не играет практически никакой роли. Почти все системные службы запускаются в конфигурационных файлах Upstart. В случае с серверами все обстоит иначе: даже в Ubuntu 12.04, самом современном LTS-варианте Ubuntu для применения на сервере, элементарные серверные службы (например, Apache, Postfix и SSH) запускаются с помощью классической Init-V.

Управляющие команды. В Upstart предусмотрено несколько новых команд. Они используются для запуска и остановки процессов, обзора текущих и предстоящих событий и т. д. Обратите внимание, что эти команды предназначены для работы только с теми процессами, которые управляются непосредственно Upstart. Службы, запускаемые через уровень совместимости Init-V, и в дальнейшим администрируются с помощью команд из Debian (в частности invoke.rc, update-rc.d и т. д.).

Параметр start или stop, соответственно, запускает или останавливает процесс, для которого в /etc/event.d предусмотрен файл конфигурации; status указывает, в каком состоянии сейчас находится процесс.

```
root# status tty2
tty2 start/running, process 4116
root# stop tty2
tty2 stop/waiting
```

Вместо start и stop можно применять и универсальную команду service, совместимую со многими другими дистрибутивами:

```
root# service tty1 stop
root# service tty1 start
root# service tty1 restart
```

В зависимости от того, какие команды были переданы initctl в виде параметров, сценарий выполняет различные задачи по администрированию системы (см. man initctl). Например, initctl emit eventname создает событие с указанным именем. Такой метод очень удобен при тестировании написанных вами сценариев. Команда initclt list выдает обзор статусов всех действующих в данный момент процессов:

```
root# initctl list | sort
acpid start/running, process 751
alsa-restore stop/waiting
```

```
alsa-store stop/waiting
anacron stop/waiting
apport stop/waiting
atd start/running, process 753
...
```

Уровень запуска. Как уже было сказано выше, Upstart не нуждается в уровнях запуска. Однако ради совместимости с процессом Init-V Upstart моделирует уровни запуска по образцу Init-V. Команда `runlevel` выдает справку о действующем уровне запуска; `telinit` *n* или `init` *n* активизирует новый уровень запуска *n*.

Специфические особенности Fedora и RHEL. Хотя в версиях Fedora 9–13 и в RHEL 6 для запуска системы применялась программа Upstart, стандартный уровень запуска в этих дистрибутивах по-прежнему настраивается в `/etc/inittab`. Но все остальные настройки, содержащиеся в этом файле, игнорируются! X использует консоль 1 (в не консоль 7, как во многих других дистрибутивах). Тем не менее, в этой консоли на уровне запуска `5` *не открывается* окно терминала.

Если во время инициализации системы была нажата клавиша I, то `touch` создает файл `/var/run/confirm`; существование этого файла проверяется в `/etc/rc.d/rc`. Если он существует, то после выполнения каждого из сценарных файлов `rc`*n*`.d`-система выдает запрос Yes/No/Continue (Да/Нет/Продолжить), где «Продолжить» означает, что остальные сценарные файлы должны выполняться без такого запроса. Интерактивный режим удобен в тех случаях, когда при выполнении сценария Init-V для активизации уровня запуска возникают проблемы. Но учитывайте, что интерактивный режим начинает работать только после завершения инициализации системы.

Файл `/etc/rc.d/rc.local` позволяет без труда индивидуально настроить процесс Init-V. Этот сценарий выполняется после всех остальных сценариев Init-V, если активизирован уровень запуска `2`, `3`, `4` или `5`. При дальнейших сменах уровня запуска или при перезагрузке компьютера этот сценарий уже не выполняется! Данный файл учитывается и в современных версиях Fedora, использующих Systemd.

16.3. Systemd

Systemd — это новая система Init, разработанная сотрудником Red Hat Леннартом Поттерингом и впервые примененная в Fedora 15. К использованию Systemd перешел и дистрибутив openSUSE, начиная с версии 12.1. В Debian версии 7 также присутствуют пакеты с Systemd, но по умолчанию Debian 7 продолжает использовать процесс Init-V.

Пожалуй, важнейшее отличие между традиционной системой Init-V и Systemd заключается в том, что за конфигурацию новой системы отвечают не оболочковые сценарии, а обычные текстовые файлы. Сам Systemd является скомпилированной программой и поэтому работает значительно быстрее своих аналогов. (В ходе обычного старта с применением Init-V запускается множество экземпляров оболочки — а на это, конечно же, уходит масса времени и ресурсов.)

В дальнейшем Systemd запускает службы параллельно, что особенно удобно при работе на многоядерных системах. Файл Upstart, а также различные варианты

Init-V, также используемые, например, в ipenSUSE до версии 11.4 либо в Debian, тоже допускают такую параллельную обработку.

Systemd использует Cgroups для выполнения и отслеживания процессов. Сокращение Cgroups означает Control Groups (Контрольные группы). Контрольные группы — это функция ядра, доступная в версии 2.6.24, которая ограничивает ресурсы процесса (процессорное время, память, ввод/вывод). Systemd запускает каждый процесс в предназначенной ему контрольной группе (Cgroup). Если количество процессов в конкретной группе снижается до 0, то Systemd понимает, что процесс был завершен (возможно, аварийно), и при необходимости может запустить его снова.

Администрирование. Ключевая команда для администрирования `systemd` называется `systemctl`. С ее помощью можно вручную запускать, останавливать и т. д. службы, описанные в файле `*.service`.

```
root# systemctl start   ntpd.service     (Запуск NTP-демона)
root# systemctl stop    ntpd.service     (Остановка NTP-демона)
root# systemctl restart ntpd.service     (Перезапуск NTP-демона)
root# systemctl reload  ntpd.service     (Новое считывание конфигурации
                                         NTP-демона)
root# systemctl status  ntpd.service     (Определение состояния
                                         NTP-демона)
```

`systemctl` можно применять и для того, чтобы надолго активизировать или деактивировать определенную службу (как `chkconfig xxx on/off`):

```
root# systemctl enable ntpd.service  (автоматически запустить NTP-демон в будущем)
   ln -s '/lib/systemd/system/ntpd.service'
      '/etc/systemd/system/multi-user.target.wants/ntpd.service'
root# systemctl disable ntpd.service  (Больше не запускать
                                      NTP-демон автоматически)
   rm '/etc/systemd/system/multi-user.target.wants/ntpd.service'
```

Итак, при активизации демона создается новая ссылка, а при деактивации эта ссылка удаляется. Как минимум в этом отношении сходство новой системы с Init-V очевидно.

Если `systemctl` вызывается без дополнительных параметров, она возвращает список всех процессов, управляемых Systemd:

```
user$ systemctl
UNIT               LOAD   ACTIVE SUB              JOB DESCRIPTION
abrt-ccpp.service  loaded active exited Install   ABRT coredump hook
abrtd.service      loaded active running          ABRT Automated Bug
                                                  Reporting Tool
abrt-oops.service  loaded active running          ABRT kernel log watcher
abrt-vmcore.service   loaded active exited        Harvest vmcores for ABRT
...
121 units listed. Pass --all to see inactive units, too.
```

Уровни запуска. В Systemd имеются компоненты, аналогичные уровням запуска. Они называются target (цель). Для обеспечения совместимости существуют специальные цели, которые соответствуют обычным уровням запуска.

Например, уровню запуска 0 соответствует oweroff.target, а уровню запуска 5 — graphical.target.

В отличие от ситуации с системой Init-V, в Systemd могут одновременно быть активны несколько целей. Список всех целей выводит следующая команда. Из вывода понятно, что система находится в состоянии, соответствующем уровню запуска 5 (активна цель graphical.target).

```
user$ systemctl list-units --type=target
UNIT                   LOAD   ACTIVE SUB     JOB DESCRIPTION
basic.target           loaded active active  Basic System
cryptsetup.target      loaded active active  Encrypted Volumes
getty.target           loaded active active  Login Prompts
graphical.target       loaded active active  Graphical Interface
local-fs-pre.target    loaded active active  Local File Systems (Pre)
local-fs.target        loaded active active  Local File Systems
multi-user.target      loaded active active  Multi-User
network.target         loaded active active  Network
remote-fs.target       loaded active active  Remote File Systems
sockets.target         loaded active active  Sockets
sound.target           loaded active active  Sound Card
swap.target            loaded active active  Swap
sysinit.target         loaded active active  System Initialization
syslog.target          loaded active active  Syslog

LOAD      = Reflects whether the unit definition was properly loaded.
ACTIVE    = The high-level unit activation state, i.e. generalization of SUB.
SUB       = The low-level unit activation state, values depend on unit type.
JOB       = Pending job for the unit.

14 units listed. Pass --all to see inactive units, too.
```

Чтобы изменить текущий уровень запуска, выполните следующую команду:

```
root# systemctl isolate reboot.target (Перезапуск компьютера)
```

Стандартный уровень запуска задается ссылкой /etc/systemd/system/default.target. Следующая команда определяет, что в будущем не будет больше запускаться графический интерфейс, а будет активизироваться только многопользовательский модуль (в соответствии с уровнем запуска 3):

```
root# ln -sf /lib/systemd/system/multi-user.target
      /etc/systemd/system/default.target
```

Конфигурация. Конфигурационные файлы systemd находятся в каталогах /etc/systemd/ и /lib/systemd/. Например, в Fedora 17 общее количество файлов в этих каталогах составляет около 250. Таким образом, конфигурация довольно сложна и не менее запутана, чем в Init-V.

Одну из важнейших ролей в концепции Systemd играют так называемые единицы (Units). Единицы — это объекты, которые должны управляться системой инициализации. К единицам относятся не только службы, которые требуется запускать и останавливать, но и сетевые интерфейсы, каталоги для подключения к системе (монтирования), разделы подкачки и т. д.

С помощью *.target-файлов можно объединять несколько единиц в группу. Принципиально цели схожи с уровнями запуска, но, как правило, целей существует гораздо больше, чем уровней запуска, и цели могут быть взаимосвязаны друг с другом. Каждая цель может быть соединена с дополнительным каталогом *имя*.target.wants, где перечисляются другие единицы, которые требуется активировать — в виде файлов или ссылок на файлы. Например, в каталоге /lib/systemd/system/sysinit.target.wants содержатся различные ссылки на файлы *.service, предназначенные для инициализации системы.

*.service-файлы описывают, какие предварительные условия должны быть выполнены для запуска службы и какая команда должна запускаться. Эти файлы находятся в каталогах /etc/systemd/system и /lib/systemd/system. В качестве примера рассмотрим следующий листинг с файлом httpd.service, который отвечает в дистрибутиве Fedora за запуск веб-сервера Apache.

```
# Файл /lib/systemd/system/httpds.service

[Unit]
Description=The Apache HTTP Server (prefork MPM)
After=syslog.target network.target

[Service]
Type=forking
PIDFile=/var/run/httpd/httpd.pid
EnvironmentFile=/etc/sysconfig/httpd
ExecStart=/usr/sbin/httpd $OPTIONS -k start
ExecReload=/usr/sbin/httpd $OPTIONS -t
ExecReload=/bin/kill -HUP $MAINPID
ExecStop=/usr/sbin/httpd $OPTIONS -k stop

[Install]
WantedBy=multi-user.target
```

Для конфигурации и, в первую очередь, для анализа Systemd вы также можете воспользоваться относительно простой программой с пользовательским интерфейсом, которая называется systemadm (рис. 16.1). В Fedora эта программа находится в пакете systemd-gtk.

Зависимости. Анализировать вручную все зависимости, существующие между службами и целями, довольно сложно. Но можно воспользоваться командой systemctl dot, которая выстраивает специальные информационные диаграммы. Такие диаграммы с помощью команды dot из пакета graphvic можно преобразовывать в файлы формата SVG (масштабируемая векторная графика). Затем эти файлы можно просмотреть в Firefox или Inkscape. Полученный в результате график можно распечатать в виде постера, повесить на стену и выдавать за произведение современного искусства. На рис. 16.2 показано менее процента такого графика!

```
user$ systemctl dot | dot -Tsvg > ~/systemd.svg
```

Реагирование на Ctrl+Alt+Delete. Если при работе в текстовой консоли нажать сочетание клавиш Ctrl+Alt+Delete, то произойдет перезапуск компьютера. За это отвечает специальный файл цели /lib/systemd/system/ctrl-alt-del.target.

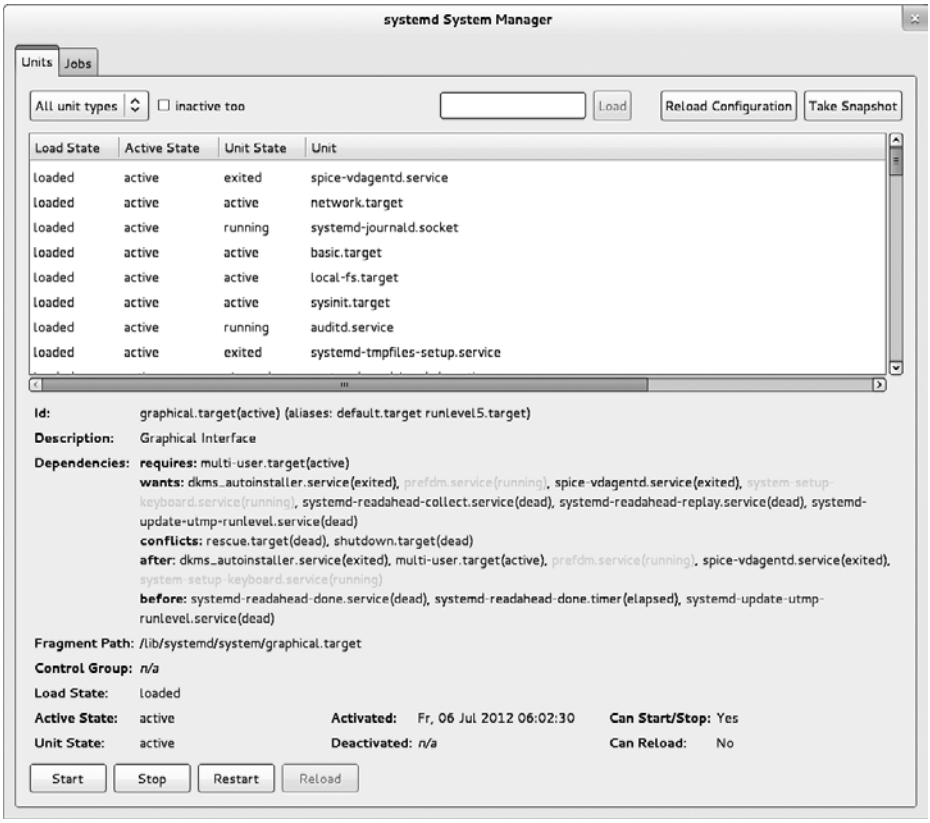


Рис. 16.1. Администрирование Systemd с помощью systemadm

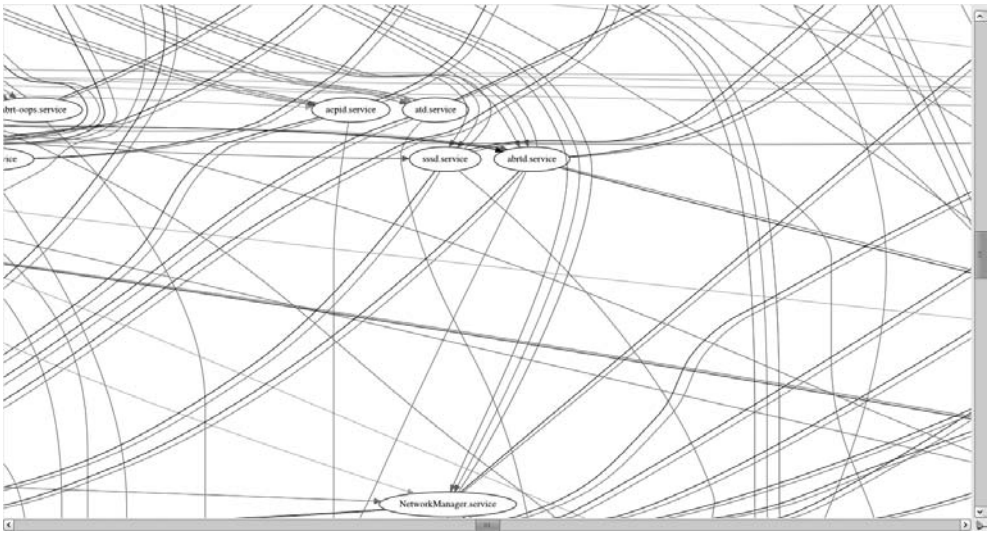


Рис. 16.2. Диаграмма зависимостей, существующих между целями и службами в Systemd

Унификация конфигурационных файлов. В процессе постепенного перехода на Systemd его разработчик Леннарт Поттеринг предложил также унифицировать различные конфигурационные файлы, которые (практически) во всех дистрибутивах хранятся в собственном месте. Кроме того, их синтаксис подчиняется частично несхожим правилам. Поскольку пока эти рекомендации не соблюдаются даже в Fedora, я не буду рассказывать здесь о документации по планируемым новым местам хранения этой информации.

Каталог /run. Systemd сохраняет идентификаторы запущенных процессов в каталоге `/run`, а не в каталоге `/var/run`, применявшемся ранее для этой цели. Содержимое `/run` физически не хранится на жестком диске, а находится только в оперативной памяти. Поэтому каталог `/run` может быть создан уже в начале процесса инициализации, независимо от того, где впоследствии будет физически находиться каталог `/var`.

Ради обеспечения совместимости каталог `/var/run` продолжает использоваться, но он представляет собой всего лишь дополнительную точку для подключения к системе в каталоге `/run` (то есть `mount --bind /run /var/run`). Каталог `/var/lock` также в действительности находится в `/run/lock`. Другие современные дистрибутивы тоже переходят от `/var/run` к `/run`. Такова даже Ubuntu, несмотря на то, что она применяет Upstart, а не Systemd. Технически любознательные пользователи Linux могут почитать о причинах такого решения на сайте http://thread.gmane.org/gmane.linux.redhat.fedora.devel/146976.

Совместимость. Systemd совместим с традиционной системой Init-V. Сценарии Init, находящиеся в каталоге `/etc/init.d/`, запускаются и останавливаются, как и раньше. В то время как Fedora практически полностью перешла на Systemd, в openSUSE каталог `/etc/init.d/` по-прежнему содержит множество сценариев, только ожидающих портирования на Systemd.

Команда `service`, поддерживаемая уже практически во всех дистрибутивах, функционирует и в Systemd. Если, например, вы выполните `service httpd`, то `service` распознает, что веб-сервер управляется Systemd, и выполнит `systemctl stop httpd.service`.

Документация. Программа Systemd отлично документирована. Все функции, стратегии и методы, связанные с работой Systemd, предельно подробно рассмотрены в различных мануалах (в частности, в `man systemd`).

Если вы предпочитаете краткие и лаконичные тексты, вам понравятся два следующих сайта. Там обобщены важнейшие изменения, внесенные в Systemd по сравнению с Init-V, а также даются ответы на некоторые часто задаваемые вопросы:

- http://fedoraproject.org/wiki/SysVinit_to_Systemd_Cheatsheet;
- http://www.freedesktop.org/wiki/Software/systemd/FrequentlyAskedQuestions.

16.4. Запуск системы Debian

На рис. 16.3 изображено, как в Debian происходит запуск системы, основанный на Init-V. По умолчанию задан уровень запуска `2`. В табл. 16.3 указано, где находятся конфигурационные файлы.

Таблица 16.3. Конфигурация запуска системы в Debian

Функция	Конфигурационные файлы
Инициализация системы	/etc/init.d/rcS, /etc/rcS.d/*
Сценарии Init	/etc/init.d/*
Ссылки на уровни запуска	/etc/rc*n*.d/rc*n*.d/*
Конфигурационные файлы	/etc/default/*

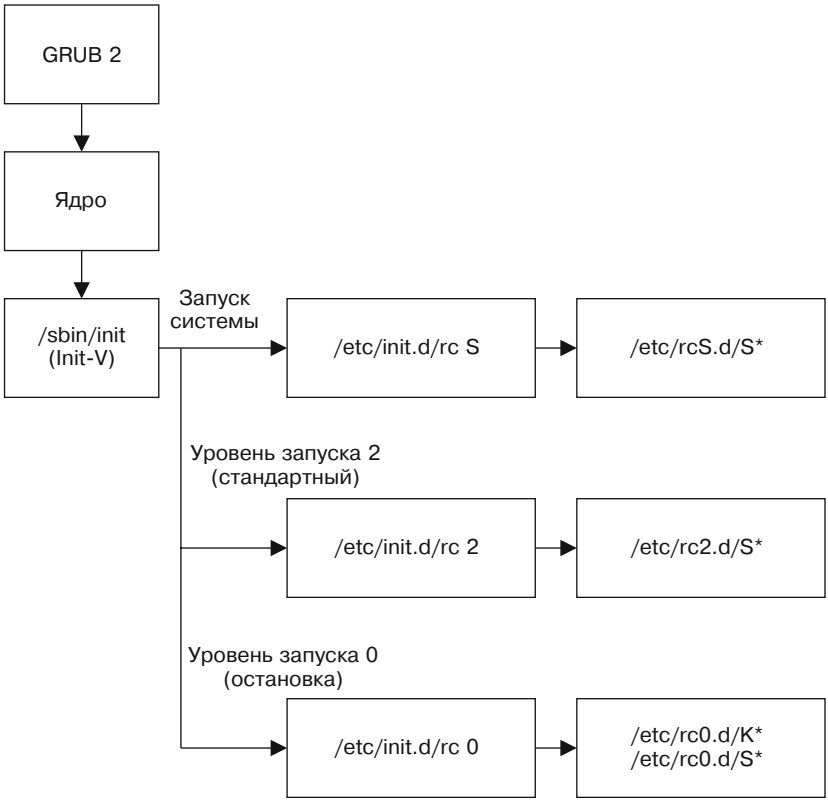


Рис. 16.3. Старт системы на уровне запуска 2 в Debian

Внутренняя организация смены уровней запуска. При переходе с одного уровня запуска на другой останавливаются только те функции, которые были активизированы на предыдущем уровне запуска, а на новом уровне запуска не нужны. Аналогичным образом запускаются только такие функции, которые до этого были неактивны. Чтобы определить, какие функции уже запущены, а какие еще нет, сценарий `rc` проверяет, есть ли для конкретной функции на предыдущем уровне запуска ссылка `start` или `stop`.

Запуск сценариев Init-V. Команда `invoke.rc` частично избавляет вас от работы с клавиатурой при вводе сценария Init-V вручную.

```
root# invoke.rc samba restart
```

Запуск X. Графическая система X запускается с помощью сценария Init-V `gdm`. Он имеет порядковый номер и стартует после всех сетевых демонов.

Параллельное выполнение сценариев Init-V. Система Init-V дистрибутива Debian способна параллельно запускать сценарии Init-V с одинаковым порядковым номером. Это помогает немного повысить эффективность работы, особенно при использовании компьютеров, на которых установлено несколько процессоров или многоядерный процессор. Ради обеспечения стабильности системы эта функция обычно не активизируется. Если вы хотите протестировать распараллеливание, используйте следующую настройку:

```
# в /etc/init.d/rc
...
CONCURRENCY=shell
```

Индивидуальная настройка процесса Init-V. С помощью файла `/etc/init.d/rc.local` вы можете индивидуально настроить процесс Init-V. Это сценарий, запускаемый после всех остальных сценариев Init-V в ситуации, когда впервые активизируется уровень запуска `2`, `3`, `4` или `5`. При последующем изменении уровня запуска или при окончании работы компьютера сценарий больше *не* выполняется (в том числе для остановки каких-либо запущенных им программ)!

Управление ссылками Init-V (insserv). В Debian начиная с версии 6 (а в SUSE — уже несколько лет) применяется команда `insserv`.

```
root# insserv samba      (Создание ссылок для запуска и остановки)
root# insserv -r samba   (Удаление ссылок для запуска и остановки)
```

Управление ссылками Init-V (update-rc.d). Команда `update-rc.d` предназначена для работы с установочными сценариями пакетов. При инсталляции или деинсталляции пакетов эта команда позволяет создавать или снова удалять ссылки на уровни запуска для сценариев Init-V конкретного пакета. В сравнительно старых версиях Debian, а также в Ubuntu вы можете использовать эту команду интерактивно. Работать с ней довольно непросто. В частности, обратите внимание на то, что `update-rc.d`, как правило, не вносит никаких изменений в уже имеющиеся ссылки! Сначала нужно удалить имеющиеся ссылки.

Команда `update-rc.d` *`имя`* `remove` удаляет все ссылки, предназначенные для запуска и остановки указанной службы. В любом случае эта команда работает только при условии, что файл `/etc/init.d/`*`имя`* уже удален. Если это условие не соблюдено, примените параметр `-f`.

Команда `update-rc.d` *`имя`* `defaults` создает на всех уровнях запуска ссылки для запуска (уровни `2-5`) и остановки (уровни `0`, `1` и `6`) той или иной службы. Названия ссылок начинаются с порядкового номера 30. Если сценарий необходимо выполнить раньше или позже в процессе запуска или остановки службы, то вам придется самостоятельно указать для запуска и остановки нужные значения (в следующем примере `13` для запуска, `1` для остановки):

```
root# update-rc.d gdm defaults 20 1
  /etc/rc0.d/K01gdm -> ../init.d/gdm
  /etc/rc1.d/K01gdm -> ../init.d/gdm
  /etc/rc6.d/K01gdm -> ../init.d/gdm
```

```
  /etc/rc2.d/S20gdm -> ../init.d/gdm
  /etc/rc3.d/S20gdm -> ../init.d/gdm
  /etc/rc4.d/S20gdm -> ../init.d/gdm
  /etc/rc5.d/S20gdm -> ../init.d/gdm
```

Если нужно отдельно настроить ссылки на каждый уровень запуска, передайте аргументы `update-rc.d` в форме *имя* `start|stop` *nn* `runlevel`. При этом *nn* — это число, стоящее в начале ссылки на уровень запуска. Можно указывать несколько уровней запуска и групп аргументов. Каждая группа должна завершаться точкой. Следующая команда действует так же, как и `gdm defaults 30 1`.

```
root# update-rc.d gdm start 30 2 3 4 5 . stop 1 0 1 6 .
Adding system startup for /etc/init.d/gdm ...
  /etc/rc0.d/K01gdm -> ../init.d/gdm
  /etc/rc1.d/K01gdm -> ../init.d/gdm
  /etc/rc6.d/K01gdm -> ../init.d/gdm
  /etc/rc2.d/S30gdm -> ../init.d/gdm
  /etc/rc3.d/S30gdm -> ../init.d/gdm
  /etc/rc4.d/S30gdm -> ../init.d/gdm
  /etc/rc5.d/S30gdm -> ../init.d/gdm
```

Если вы работаете с X, то можете настроить ссылки на уровни запуска с помощью программы Gnome `services-admin`, все изменения касаются стандартного уровня запуска.

Как построены файлы сценариев Init-V. В следующих строках показано, как построен пользовательский сценарий Init-V `/etc/init.d/masquerading`, необходимый для того, чтобы компьютер, на котором он установлен, работал как интернет-шлюз (выполнял функции маскарадинга и простейшего брандмауэра). В блоке `INIT-INFO` указано, на каких уровнях запуска обычно выполняется данный сценарий и какие другие службы должны быть предварительно запущены перед его выполнением.

```
#!/bin/sh

### BEGIN INIT INFO
# Provides:           masquerading
# Required-Start:     $network $local_fs $remote_fs
# Required-Stop:      $network $local_fs $remote_fs
# Default-Start:      2 3 4 5
# Default-Stop:       0 1 6
# Short-Description:  start masquerading
### END INIT INFO
DESC="masquerading"        # Обозначение сценария
ADSL=eth1                  # Интерфейс, через который осуществляется доступ
                           # в Интернет
. /lib/lsb/init-functions  # Считывание основных функций
IPT=$(which iptables)      # Поиск команды iptables
if [ -z $IPT ]; then
  [ -x /sbin/iptables ]     && IPT=/sbin/iptables
  [ -x /usr/sbin/iptables ] && IPT=/usr/sbin/iptables
fi
[ -z $IPT ] && (echo "iptables cannot be found!"; exit 0)
```

```
# Функции для start, stop и restart
case "$1" in
  start)
    log_begin_msg "Starting masquerading ..."
    ERROR=0
    $IPT -t nat -A POSTROUTING -o $ADSL -j MASQUERADE
    echo 1 > /proc/sys/net/ipv4/ip_forward
    log_end_msg $ERROR
    ;;

  stop)
    log_begin_msg "Stopping masquerading ..."
    ERROR=0
    echo 0 > /proc/sys/net/ipv4/ip_forward
    $IPT -t nat -D POSTROUTING -o $ADSL -j MASQUERADE
    log_end_msg $ERROR
    ;;

  restart)
    $0 stop
    $0 start
    ;;
  *)
    log_success_msg "Usage: masquerading {start|stop|restart}"
    exit 1
    ;;
esac
exit 0
```

Чтобы в дальнейшем сценарий автоматически запускался при старте компьютера, выполните следующие команды:

```
root# insserv masquerading
```

16.5. Запуск системы в Fedora

На рис. 16.4 показано, какие сценарии выполняются на стандартном уровне запуска 5 при старте системы. В табл. 16.4 перечислены конфигурационные файлы, задействованные в процессе. Эти данные касаются дистрибутива Fedora 17 с системой Systemd. Обратите внимание, что Systemd используется, начиная с Fedora 15. В более ранних версиях Fedora, таких как RHEL 6, запуск системы управляется с помощью Upstart, в RHEL 5 — с использованием классических систем Init-V.

Переход на Systemd в Fedora был очень последовательным. Каталог `/etc/init.d/` практически пуст, а сценарий `/etc/rc.d/rcsysconfig`, ранее отвечавший за инициализацию системы и оборудования, вообще исчез.

Запуск системы X. За запуск системы X отвечает конфигурационный файл `/lib/systemd/system/prefdm.service`. В файле `/lib/systemd/system/graphical.target.wants` содержится ссылка на `display-manager.service`, который, в свою очередь, является ссылкой на `prefdm.service`. То, какой именно дисплейный менеджер будет задей-

ствован, определяется в файле /etc/sysconfig/desktop. Заданная там переменная DISPLAYMANAGER интерпретируется сценарием /etc/X11/prefdm.

Таблица 16.4. Конфигурация Fedora при запуске системы

Функция	Конфигурационные файлы
Systemd	/etc/systemd/*, /lib/systemd/*
Стандартная цель Systemd	/etc/systemd/system/default.target
Инициализация системы	/lib/systemd/system/sysinit.target*
Традиционные Init-сценарии	/etc/rc.d/init.d/*
Ссылки на уровни запуска Init-V	/etc/rc.d/rcn.d/*
Конфигурационные файлы	/etc/sysconfig/*

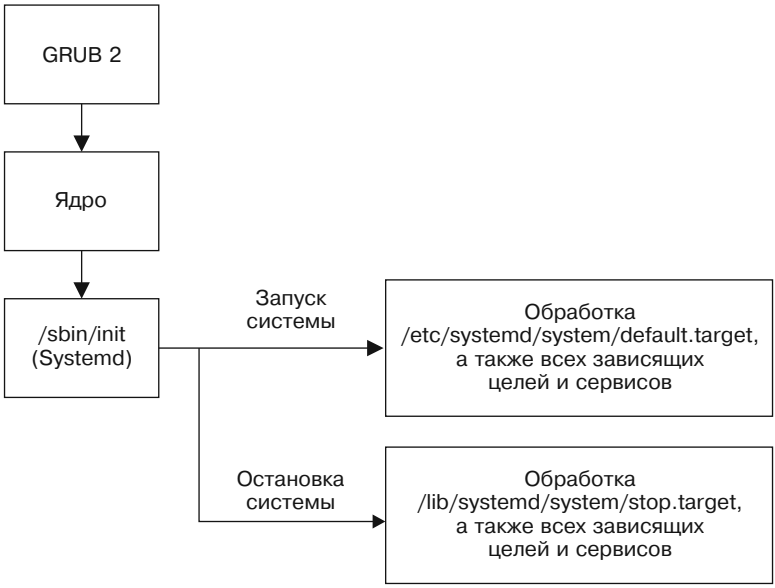


Рис. 16.4. Старт системы на уровне запуска 5 в дистрибутиве Fedora и openSUSE

Логирование (журналирование). Протокол всех служб, запускаемых systemd, находится в файле логов /var/log/boot.log. В этом файле регистрируются все сценарии Init-V, помеченные ключевым словом LSB — вероятно, чтобы указать на следование рекомендациям стандарта Linux Standard Base.

Конфигурация. Службы Systemd можно запускать, останавливать и перезапускать не только с помощью команды systemctl, но и с помощью программы с графическим пользовательским интерфейсом system-config-services. Ранее в этой программе были и другие функции, например для активизации и деактивации служб. Но они исчезли после перехода на Systemd.

Совместимость с системой Init-V. Systemd совместим с системой Init-V и поэтому отвечает за выполнение сценариев из /etc/rc*n*.d. Команда chkconfig, специфичная для Fedora и Red Hat, также помогает при ассоциировании ссылок со сценариями Init-V.

Файл /etc/rc.d/rc.local, ранее предназначавшийся для индивидуальной настройки запуска системы, в Fedora 16 уже не существует. Если в конце старта системы вы хотите запускать собственные сценарии, то для этого по-прежнему можно использовать /etc/rc.d/rc.local. Нужно создать данный файл и с помощью chmod ug+x гарантировать, что он также является исполняемым.

16.6. Запуск системы в SUSE

Поскольку дистрибутив openSUSE также перешел к использованию Systemd, рис. 16.4 отражает ситуацию и в современных версиях этого дистрибутива. В табл. 16.5 перечислены конфигурационные файлы, необходимые для запуска системы в openSUSE.

Таблица 16.5. Конфигурация запуска системы в SUSE

Функция	Конфигурационные файлы
Systemd	/etc/systemd/*, /lib/systemd/*
Стандартная цель Systemd	/etc/systemd/system/default.target
Инициализация системы	/lib/systemd/system/sysinit.target*
Традиционные Init-сценарии	/etc/rc.d/init.d/*
Ссылки на уровни запуска Init-V	/etc/rc.d/rcn.d/*
Конфигурационные файлы	/etc/sysconfig/*

Запуск X. Графическая система X запускается сценарием Init-V xdm. Какой экранный менеджер будет запущен далее (например, kdm4), определяет переменная DISPLAYMANAGER, настраиваемая в /etc/sysconfig/displaymanager.

Совместимость с Init-V. openSUSE не настолько преуспел в портировании запуска системы на Systemd, насколько Fedora: многочисленные службы по-прежнему управляются сценариями Init-V.

Традиционные сценарии Init-V в дистрибутиве openSUSE по-прежнему можно вызывать не только с помощью довольно неудобной команды /etc/init.d/name, но и в виде rcname (то есть для запуска или остановки сервера Samba используется команда rcsmb). Соответствующие ссылки находятся в /usr/sbin.

Insserv. Чтобы ссылки из /etc/init.d/rc*n*.d указывали на новый сценарий Init-V, просто выполните команду insserv scriptname. Команда insserv интерпретирует комментарии Default-Start и Default-Stop в файле сценария. Они указывают, на каком уровне запуска должен выполняться сценарий.

```
root# insserv squid
```

Для того чтобы снова удалить ссылки на сценарий, вызовите insserv с параметром -r.

Команда insserv отвечает за правильную нумерацию ссылок (нумерация имеет определяющее значение при задании порядка, в котором выполняются сценарии Init-V). Чтобы решить, какой номер будет присвоен ссылке, интерпретируются комментарии Provides и Requires в файле сценария Init-V. При необходимости insserv может пронумеровать все уже имеющиеся ссылки, чтобы можно было пра-

вильно расположить новый сценарий в ряду имеющихся. Такой автоматизм работает хорошо, но оказывается, что довольно сложно вручную повлиять на стартовую последовательность исполнения сценариев.

Chkconfig. Команда `chkconfig` присутствует в SUSE, пожалуй, именно для обеспечения совместимости. Параметры `-add`, `-del` и `-list` функционируют так же, как и в Red Hat, но с другими параметрами есть некоторые различия. Внутрисистемно `chkconfig` обращается к `insserv`.

Параллельное выполнение сценариев Init-V. Сценарии Init-V, не зависящие друг от друга, по умолчанию выполняются параллельно, а не последовательно. Управление деталями процесса загрузки осуществляется в файле `/etc/sysconfig/boot`. Чтобы обеспечить параллельное выполнение, ссылки Init-V нужно изменить с помощью команды `insserv` (но не напрямую). Она создает файлы с расширением `.depend.*`, содержащие информацию о взаимозависимостях между сценариями Init-V. Более подробно запуск системы SUSE описан на странице сайта-руководства, посвященной `init.d`.

Rc.local. В файле `/etc/rc.d/boot.local` можно выполнять локальные настройки. В этом сценарии должны содержаться только такие команды, которые однократно выполняются при запуске системы. Типичным примером подобных команд являются `modprobe`, предназначенные для загрузки конкретных модулей ядра. `Boot.local` выполняется до сценариев `rc`.

16.7. Запуск системы Ubuntu

На рис. 16.5 показано, как происходит запуск системы в дистрибутиве Ubuntu 12.10 с применением Upstart. Стандартный уровень запуска — `2`. В табл. 16.6 показано, какие конфигурационные файлы отвечают за запуск системы.

Таблица 16.6. Конфигурация запуска системы Ubuntu

Функция	Конфигурационные файлы
Upstart	/etc/init/*.conf
Инициализация системы	/etc/init.d/rcS, /etc/rcS.d/*
Init-сценарии	/etc/init.d/*
Ссылки на уровни запуска	/etc/rcn.d/rcn.d/*
Конфигурационные файлы	/etc/default/*

При запуске Ubuntu 12.10 Upstart выполняет большинство системных и сетевых служб. Остальные функции инициализации производятся с помощью сценариев, совместимых с Init-V. При этом та часть запуска, которая выполняется Init-V, протекает так же, как в Debian. Применяются те же инструменты администрирования. Стандартный уровень запуска устанавливается в файле `/etc/init/rc-sysinit.conf`.

Запуск X. Графическая система X запускается Upstart (см. файл `/etc/init/lightdm.conf`). Чтобы можно было запустить X, предварительно необходимо лишь инициализировать файловую систему и запустить систему DBUS. Экранный менеджер стартует уже через несколько секунд после начала процесса загрузки.

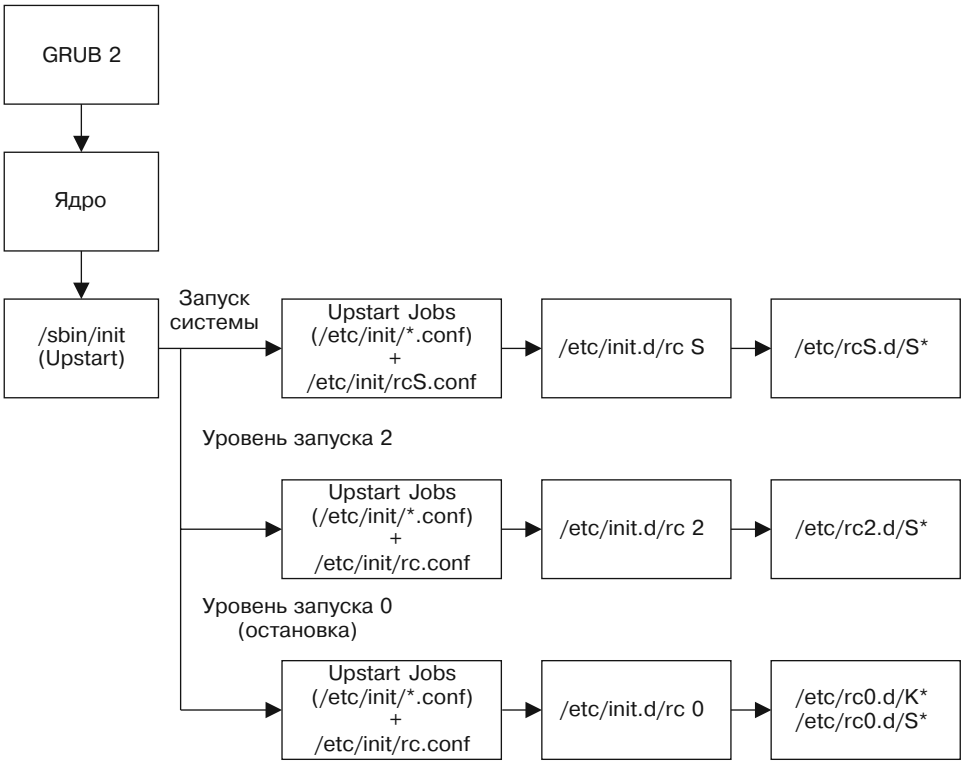


Рис. 16.5. Старт системы на уровне запуска 2 в Ubuntu

Собственные конфигурационные файлы Upstart. Чтобы самостоятельно запустить программу в рамках старта системы, нужно создать собственный конфигурационный файл в `/etc/init`. В следующем примере мы, соответственно, активизируем и деактивируем маскарадинг, не запуская при этом никакого фонового процесса. Для этого нет ни строки `exec`, ни общего блока `script`, зато есть два блока `pre-start script` и `post-stop script`. Функция маскарадинга включается на сетевых интерфейсах и при завершении работы компьютера снова выключается.

```
# Файл /etc/init/masquerading.conf
description     "masquerading"
start on (starting network-interface
          or starting network-manager
          or starting networking)
stop on runlevel [!023456]
pre-start script
   sysctl -q -w net.ipv4.ip_forward=1
   iptables -A POSTROUTING -t nat -o eth0 -j MASQUERADE
end script
post-stop script
   iptables -t nat -D POSTROUTING -o eth0 -j MASQUERADE
   sysctl -q -w net.ipv4.ip_forward=0
end script
```

Можете протестировать конфигурационный файл с помощью команд `start masquerading` и `stop masquerading`. Обратите внимание на то, что `start` или `stop` в точности повторяют имя конфигурационного файла, но без `.conf`. Если вы получите сообщение об ошибке `unknow job` (неизвестная задача), то это, как правило, объясняется наличием синтаксических ошибок в конфигурационном файле. Чтобы проверить синтаксис конфигурационного файла, выполните команду `init-checkconf`:

```
user$ init-checkconf /etc/init/masquerading.conf
File masquerading.conf: syntax ok
```

16.8. Демон интернет-сервисов

Программы, предоставляющие в распоряжение сервисы локальных сетей и Интернета, можно разделить на две группы.

- К первой относятся программы-демоны, работающие без перерывов. Почти все сетевые службы, упоминаемые в нашей книге — веб-сервер Apache, файловый сервер Samba, SSH-сервер, — относятся к этой группе. Такие программы запускаются при старте системы сценарием Init-V (а в будущем, возможно, будут запускаться Upstart) и наблюдают за работой IP-порта. Как только им встречается IP-пакет, адресованный этому порту, такой пакет интерпретируется и на него отсылается ответ.
- Во вторую входят редко используемые программы, запускаемые лишь при необходимости. Чтобы не запускать все эти программы сразу, выполняется так называемый демон интернет-сервисов. Эта программа одновременно наблюдает за несколькими IP-портами и только при необходимости активизирует соответствующие серверные службы. Сам демон интернет-сервисов запускается обычным образом — сценарием Init-V.

Ранее наиболее распространенным демоном интернет-сервисов был `inetd`. В настоящее время эта программа считается устаревшей. В зависимости от дистрибутива вместо нее могут использоваться `openbsd-inetd` или `xinetd`. Обычно ни один из этих пакетов не устанавливается по умолчанию. Инсталлировать их требуется лишь тогда, когда другой программе для работы необходим функционал `inetd`, а в пакете задана соответствующая взаимосвязь (например, SWAT).

Файл /etc/services. Независимо от того, какой именно демон интернет-сервисов применяется в вашем дистрибутиве, в файле `/etc/services` соотносятся названия различных интернет-сервисов (например, FTP, Telnet и т. д.), типы их протоколов и номера портов. Например, почтовые серверы (почтовые агенты) при работе используют порт 25 и протоколы TCP и UDP. Далее приведен фрагмент такого файла:

```
# /etc/services (выдержка)
# имя    порт/протокол псевдоним комментарий
ftp-data 20/tcp                 # Передача файлов [данные по умолчанию]
ftp-data 20/udp                 # Передача файлов [данные по умолчанию]
ftp      21/tcp                 # Передача файлов [контроль]
ssh      22/tcp                 # Протокол SSH для удаленного входа
                                # в систему
```

```
ssh      22/udp                  # Протокол SSH для удаленного входа
                                 # в систему
smtp     25/tcp        mail      # Простой протокол электронной почты
...
```

Файл /etc/inetd.conf. Если в вашем дистрибутиве работает `openbsd-inetd`, его конфигурация задается в файле `/etc/inetd.conf`. Программа запускается в том случае, если в конфигурационном файле `inetd.conf` есть как минимум одна активная запись. Каждая запись в этом файле состоит из строки с шестью столбцами.

- В первом столбце указывается название службы, которая должна быть определена в `/etc/services`.
- Во втором и третьем столбцах описано, как служба обменивается информацией (тип сокета и протокол).
- В четвертом столбце указано, должна ли одна и та же служба многократно запускаться при поступлении запросов (`nowait`) или же такие запросы должны обрабатываться только после того, как уже запущенная служба выполнит свою задачу (`wait`). В данном случае можно указать время задержки в секундах.
- В пятом столбце определено, с какими правами должен запускаться процесс.
- Остаток строки — это команда, которую требуется выполнить. При этом `tcpd` сначала проверяет, разрешено ли выполнение этой команды правилами, установленными программой TCP-Wrapper.

```
# Файл /etc/inetd.conf
swat stream tcp nowait.400 root /usr/sbin/tcpd /usr/sbin/swat
...
```

Как правило, вам не придется самостоятельно заниматься конфигурацией `inetd.conf`. При установке пакета, зависящего от `inetd`, файл `inetd.conf` автоматически дополняется соответствующим образом. Обратите внимание, что строки, начинающиеся с #, как обычно, считаются комментариями! Изменения, внесенные в `inetd.conf`, вступают в силу только после выполнения `/etc/init.d/openbsd-inetd reload`.

Файл /etc/xinetd.conf. В файле `/etc/xinetd.conf` содержатся некоторые базовые настройки `xinetd`, касающиеся, например, функции журналирования и стандартного IP-адреса. Как правило, настройки можно оставить без изменений. Очень важна команда `includedir`, указывающая каталог с остальными конфигурационными файлами (обычно `/etc/xinetd.d`).

Каталог xinetd.d/*. В каталоге `/etc/xinetd.d` содержатся отдельные конфигурационные файлы для каждой службы, управляемой `xinetd`. Названия файлов в `/etc/xinetd.d` не играют роли: `xinetd` просто считывает все файлы из этого каталога и интерпретирует их (не учитываются файлы, названия которых заканчиваются символом ~ или содержат точку).

Отдельные конфигурационные файлы построены одинаково. В следующем примере показан файл для сервера RSYNC:

```
# /etc/xinetd.d/rsync
service rsync
{
```

```
        disable         = yes
        socket_type     = stream
        wait            = no
        user            = root
        server          = /usr/bin/rsync
        server_args     = --daemon
        log_on_failure += USERID
}
```

Кратко расскажу о важнейших ключевых словах, которые могут встречаться в конфигурационных файлах `xinetd`. Подробное описание дается в `man xinetd.conf`.

- `service` — означает службу (в соответствии с `/etc/services`).
- `socket_type` и `protocol` — указывают, как должны передаваться файлы между клиентом и сервером.
- `type=INTERNAL` — означает, что данная служба предоставляется `xinetd` напрямую.
- `server` — задает имя программы (если это не внутренняя служба `xinetd`).
- `server_args` — указывает необязательные параметры, которые можно сообщить службе при запуске.
- `user` — говорит, под какой учетной записью выполняется программа (обычно root, но это может быть news, mail и т. д.).
- `disable=Yes/No` — указывает, активна служба или заблокирована. Если служба заблокирована, то в конфигурационном файле стоит `disable=Yes`. В Fedora и Red Hat службы xinetd также можно деактивировать с помощью команды `chkconfig --del` *имя* и снова активизировать, используя `chkconfig --add` *имя*. Эти команды изменяют только строку `disable`, не затрагивая остальную конфигурацию.
- `log_*` — указывает, следует ли протоколировать использование службы.

Файлы /etc/hosts.allow и hosts.deny. Определяют, с какого компьютера и какие службы могут использоваться. Настройки действительны лишь для тех программ, которые при работе обращаются к библиотеке TCP-Wrapper. К таким программам, кроме `xinetd`, также относятся сервер SSH, NFS и программа CUPS в SUSE. Строение конфигурационных файлов `hosts.allow` и `hosts.deny` подробно описано в пункте «Файлы /etc/hosts.allow и hosts.deny» подразделе «Библиотека TCP-Wrapper» раздела 26.2.

Особый случай представляет собой `openbsd-inetd`: этот пакет связан с библиотекой TCP-Wrapper, но по умолчанию ее функции отключены! Чтобы активизировать их, нужно запустить `openbsd-inetd` с параметром `-l`. В Debian и Ubuntu для этого создается файл `/etc/default/openbsd-inetd`, в который вставляется следующая строка:

```
# Файл /etc/default/openbsd-inetd (Debian, Ubuntu)
# Активизация функций TCP-Wrapper
OPTIONS="-l"
```

С таким параметром `inetd.conf` должен еще вызвать `tcpd`, чтобы избежать повторной проверки правил TCP-Wrapper.

17 Ядро и модули

В этой главе будет рассмотрено ядро Linux и его модули. Модули — это части ядра, которые можно загружать по мере необходимости, именно тогда, когда впервые запрашивается тот или иной аппаратный компонент. В разделе 17.1 рассказывается, почему такое подключение модулей обычно осуществляется автоматически и что делать, если автоматическая система откажет.

Достаточно редко возникает необходимость заново компилировать ядро. Компилировать отдельный модуль так, чтобы он подошел к актуальному ядру, приходится гораздо чаще (например, для драйвера графической карты или для VirtualBox). В разделе 17.2 доказано, что компилировать целое ядро или отдельные модули не так сложно и что эта работа по плечу простым смертным. В разделе 17.3 вы узнаете, как получить из файловой системы `/proc` или `/sys` актуальную информацию о ядре. В разделе 17.4 рассказано, как можно передавать параметры ядру во время загрузки компьютера. И, наконец, в разделе 17.5 мы поговорим о том, как изменять параметры ядра прямо во время работы компьютера.

Разумеется, эта глава предназначена только для продвинутых пользователей Linux. Начинающим я настоятельно рекомендую работать с готовым ядром системы и не устанавливать пакетов, предназначенных для использования в другом дистрибутиве! Вся информация, приведенная в данной главе, касается версий ядра как 2.6.*n*, так и 3.*n*. Хотя в середине 2011 года номер версии ядра изменился, что позволяет предположить наличие коренных изменений, не все так сложно. После перехода с 2.6.39 на 3.0 в ядре не появилось никаких принципиально новых черт кроме обычных обновлений и усовершенствований драйверов.

17.1. Модули ядра

Ядро — это часть Linux, отвечающая за выполнение простейших функций, таких как управление памятью, доступ к жестким дискам и сетевым картам и т. д. При этом ядро организовано по модульному принципу: сначала, то есть при запуске компьютера, загружается основное ядро, содержащее только функции, которые требуются для старта системы.

Если в ходе работы понадобятся дополнительные функции (например, для работы с конкретным оборудованием), то необходимый код подключится к ядру как модуль. Если в течение некоторого времени эти дополнительные функции

будут не нужны, модуль может быть удален из ядра. У такой модульной концепции множество преимуществ.

- Модули можно подключать по мере необходимости. Если определенный модуль используется редко, то, не подключая его, вы можете экономить память. Это означает, что ядро имеет размер не больше, чем это необходимо, и оптимально подходит к оборудованию пользователя.
- При изменении конфигурации оборудования (например, при подключении новой сетевой карты) не нужно компилировать новое ядро — можно просто подключить соответствующий модуль. Все распространенные дистрибутивы построены по такому принципу.
- При разработке нового модуля не требуется все время перезапускать компьютер. Достаточно заново скомпилировать модуль. Затем этот модуль можно будет протестировать, не останавливая работу системы.
- Производители аппаратного обеспечения могут предоставлять модули для поддержки своего оборудования в виде двоичных файлов, не раскрывая при этом код. Пока такой метод применяется лишь иногда и, конечно же, он не лишен недостатков. (Модуль должен точно подходить к ядру, то есть для каждой новой версии ядра он должен компилироваться повторно. Если обнаружится ошибка, ее может исправить только производитель, но не пользователь — ведь исходный код недоступен.)

Подробная базовая информация по работе с модулями ядра сообщается на сайте http://www.tldp.org/HOWTO/Module-HOWTO/.

Автоматическая загрузка модулей. За то, чтобы модули ядра действительно загружались автоматически по мере необходимости, отвечает компонент kmod, интегрированный в ядро. Он управляется файлом `/etc/modprobe.conf`, о котором подробно рассказано в подразделе «Конфигурация модуля» данного раздела далее.

Ядро и модули должны подходить друг к другу. До версии 2.6.15 ядро и его модули должны были точно подходить друг к другу: невозможно было загрузить модуль, скомпилированный для другой версии ядра (пусть и очень близкой). Для каждой версии ядра создавался отдельный каталог модулей `/lib/modules/`*`версия_ядра`*. При применении как раз таких модулей, которые не поставляются вместе с дистрибутивом (таковы, в частности, модули для графических драйверов ATI или NVIDIA), становится очень сложно точно подобрать модули к ядру.

Возможность использования модулей другой версии ядра. В версии ядра 2.6.16 и выше появилась функция `module versioning`, позволяющая применять модули и из других версий ядра, и ситуация значительно упрощается: вместе с модулем сохраняется дополнительная информация, сообщающая, возможно ли совместно использовать конкретный модуль и конкретное ядро, несмотря на то, что они относятся к разным версиям. Часто можно применить и такой модуль, версия (номер) которого не соответствует версии ядра. Чтобы этот механизм работал, необходимо активизировать функцию `module versioning` уже при компилировании, кроме того, не должно быть никаких изменений в интерфейсах, расположенных между модулем и ядром.

Если вы хотите написать для SUSE собственные модули ядра, поддерживающие module versioning, сначала установите пакет kernel-syms. Функция module versioning также может называться kernel symbol versions или modversions.

Команды для управления модулями

Во всех распространенных дистрибутивах предусмотрена возможность загрузки дополнительных модулей по мере надобности. Пример: с помощью команды mount вы подключаете к дереву каталогов файловую систему USB-флешки. При этом автоматически активизируется модуль vfat, необходимый для считывания файловой системы флешки.

Итак, управление модулями, как правило, осуществляется автоматически и является прозрачным, а вам не приходится прибегать к описанным ниже командам, предназначенным для управления модулями вручную. И все же рекомендую изучить эти команды, чтобы при необходимости иметь возможность вмешаться в работу модулей.

Все модули находятся в каталоге /lib/modules/*n*, где *n* — версия действующего ядра. Файлы модулей имеют расширение *.ko.

Определение версии ядра. Команда uname -r выдает номер версии используемого ядра:

```
user$ uname -r
3.5.0-2-generic
```

Загрузка файла модуля. Команда insmod интегрирует указанный модуль в ядро. При этом необходимо передать полное имя файла. Дополнительно можно передать модулю параметры. Если вы хотите указать шестнадцатеричные значения, перед параметрами нужно ставить 0x, то есть option=0xff.

```
root# insmod /lib/modules/3.5.0-1.1-desktop/kernel/fs/fuse/fuse.ko
```

Команда insmod -f пытается загрузить модуль даже тогда, когда он не подходит к действующей версии ядра. Если между ядром и модулем нет никакой несовместимости, такая попытка будет успешной.

Как правило, для загрузки модулей ядра применяется не insmod, а modprobe. Эта команда имеет два преимущества: сама ищет файл модуля (вам следует указать только название модуля), а при необходимости также загружает все модули, требуемые для работы. Кроме того, учитываются все параметры модуля, указанные в /etc/modprobe.conf. Для работы команды modprobe необходимо, чтобы модуль имел правильную конфигурацию (modprobe.conf и modules.dep).

```
root# modprobe fuse
```

Список загруженных модулей. Команда lsmod возвращает, как правило, очень длинный список всех модулей ядра, загруженных в настоящий момент:

```
root# lsmod | sort
Module        Size   Used by
ac            4933   0
autofs4      19013   1
```

```
battery        9285   0
bluetooth     44069   5 hidp,rfcomm,l2cap
button         6609   0
...
fuse          36313   0
...
```

Удаление модулей. Команда `rmmod` вновь удаляет указанный модуль из ядра и предоставляет память, которую занимал данный модуль. Эта команда может быть выполнена лишь в том случае, если удаляемый модуль сейчас не используется. Команда `rmmod -a` удаляет из памяти все модули, не применяемые в данный момент.

```
root# rmmod fuse
```

Информация о модулях. Команда `modinfo` выдает подробную информацию о модуле. Он не должен находиться в ядре. В следующем примере приведены данные по модулю e1000. Это драйвер для сетевого адаптера Intel.

```
root# modinfo e1000
filename:    /lib/modules/3.5.0-1.1-
                desktop/kernel/drivers/net/e1000/e1000.ko
version:     7.3.21-k8-NAPI
license:     GPL
description: Intel(R) PRO/1000 Network Driver
author:      Intel Corporation, <linux.nics@intel.com>
...
depends:
vermagic:    3.5.0-1.1-desktop SMP preempt mod_unload modversions
parm:        TxDescriptors:Number of transmit descriptors (array of int)
parm:        RxDescriptors:Number of receive descriptors (array of int)
...
```

Конфигурация модуля

Управление модулями происходит как по волшебству.

- Если вы подключаете к файловой системе новый раздел и в нем используется такой формат файловой системы, который ранее не применялся, то автоматически загружается модуль для работы с этой файловой системой.
- Если раздел находится на диске SCSI, то активизируются и модули SCSI (если они еще не были загружены).
- В ходе инициализации сетевых функций автоматически загружается драйвер, необходимый для работы вашей сетевой карты и т. д.

За автоматическую загрузку модулей ядра отвечает компонент `kmod`, интегрированный в ядро. Чтобы все эти функции правильно работали, используются конфигурационные механизмы.

- **Модули, необходимые для запуска компьютера.** Некоторые модули ядра нужны уже при запуске компьютера — в частности, модули для доступа к файловой системе. Пусть они и не являются неотъемлемой частью ядра, но

должны передаваться ядру загрузчиком GRUB в файле `Initrd` при запуске компьютера.

- **Модули основных функций.** Модули для базового управления компонентами оборудования (например, для системы USB) загружаются различными сценариями процесса Init-V, которые реагируют прямо на команды `modprobe`.
- **Модули для интерфейсов.** Несколько других модулей загружается в тот момент, когда впервые используется отдельный интерфейс. Здесь возникает особая проблема — для работы с некоторыми интерфейсами конкретные аппаратные компоненты используют различные модули. Итак, если вы запрашиваете интерфейс `eth0` для первой сетевой карты в компьютере, нужно загрузить в ядро модуль, подходящий для этой карты.

  Вы должны сообщить ядру, какой модуль нужно применять. Такая информация находится в `/etc/modprobe.conf`, а также в файлах каталога `/etc/modprobe.d`. Там же располагаются установочные и характерные для дистрибутивов параметры и команды, указывающие, какие модули *не должны* загружаться автоматически (файл `blacklist`).

  Система `udev`, автоматически управляющая устройствами, также при необходимости загружает нужные модули. Соответствующие правила содержатся в файле `/etc/udev/rules.d`.
- **Модули для работы с устройствами USB, Firewire и т. д.** Такое оборудование играет особую роль. Несколько файлов `*.*map` в каталоге `/lib/modules/`*`версия_ядра`*`/` на основании идентификационных кодов компонентов оборудования определяют, какой модуль требуется загрузить.
- **Взаимосвязи между модулями.** Многие модули зависят друг от друга. Например, модуль `nfs` для файловой системы NFS функционирует лишь в том случае, если в систему также загружены модули `lockd`, `nfs_acl` и `sunrpc`. Все такие зависимости между модулями перечислены в файле `/lib/modules/n/modules.dep`.

Загрузка модулей при запуске компьютера. Иногда, независимо от описанных здесь способов конфигурации, требуется, чтобы при старте компьютера загружался определенный модуль ядра — причем не автоматически, а по вашей инициативе. Оптимальный метод в таком случае выбирается в зависимости от дистрибутива.

В Debian и Ubuntu все просто. В этих дистрибутивах специальный сценарий Init-V `/etc/init.d/module-init-tools` отвечает за то, чтобы загрузить все модули, перечисленные в `/etc/modules`, строка за строкой. Иначе говоря, вам всего лишь нужно добавить новый модуль как новую строку в `/etc/modules`.

В большинстве других дистрибутивов необходимо добавить команду `modprobe` *`имя_модуля`* в сценарий Init-V, предназначенный для локальных настроек. Однако не забывайте, что в определенных дистрибутивах модули в таком случае будут загружаться лишь в финале процесса Init-V (при выполнении некоторых задач это достаточно поздно):

- Red Hat, Fedora — `/etc/rc.d/rc.local`;
- SUSE — `/etc/init.d/boot.local`.

Синтаксис modprobe

В нескольких следующих абзацах описаны важнейшие ключевые слова для `modprobe.conf` и файлы, находящиеся в `modprobe.d/`. Этот вопрос детально рассмотрен в справке `man modprobe.conf`.

Alias. Эти команды указывают, какие модули ядра с какими устройствами используются. Пример: с устройством `/dev/eth0` должен применяться модуль 8139too.

```
alias eth0 8139too
```

Доступ ко многим аппаратным компонентам осуществляется через блочные и символьные файлы-устройства (`/dev/`*`xxx`*). С точки зрения ядра, эти файлы-устройства называются не именами, а старшим и младшим номером устройства. Многочисленные команды `alias` обеспечивают взаимосвязь между номерами устройств и модулями. Схожим образом определяются и сетевые протоколы: для использования определенного протокола ядро ищет семейство протоколов под названием net-pf-n. В следующем примере для семейства протоколов 5 загружается модуль AppleTalk.

```
alias net-pf-5 appletalk
```

Если вам не нужен этот протокол и тем более если соответствующий модуль не установлен, то следующая команда избавит вас от назойливых сообщений об ошибках:

```
alias net-pf-5 off
```

Options. Команды `options` указывают, с какими параметрами должен загружаться тот или иной модуль. Благодаря следующей команде модуль ne (совместимые с NE-2000 Ethernet-карты) загружается с параметром `io=0x300`.

```
options ne io=0x300
```

Include. Команды `include` загружают указанные конфигурационные файлы.

Install. С помощью `install` даются команды, выполняемые в дополнение к простой загрузке модуля. Я приведу пример, который ради экономии места разделен на две строки. Если вам потребуется модуль ALSA `snd`, нужно выполнить следующие команды:

```
install snd modprobe --ignore-install snd $CMDLINE_OPTS && \
  { modprobe -Qb snd-ioctl32 ; : ; }
```

Remove. С помощью `remove` указываются команды, которые должны выполняться при удалении модуля.

Blacklist. При использовании команды `blacklist` внутренние alias-определения модулей не учитываются. Как правило, команды `blacklist` находятся в файле `/etc/modprobe.d/blacklist`. Здесь расположены модули, которые *не должны* загружаться — из-за проблем с совместимостью или потому, что для них есть лучшая альтернатива. Например, следующая строка не дает запуститься модулю usbmouse.

```
blacklist usbmouse
```

Компилирование дополнительного модуля

Если вы применяете Linux вместе с VirtualBox, желаете использовать двоичные графические драйверы ATI или NVIDIA или вам нужен иной специфичный аппаратный модуль, которого нет в ядре вашего дистрибутива, вам потребуется скомпилировать модуль, подходящий к применяемой версии ядра.

Инструменты разработки. Для компилирования модулей требуются не только компиляторы с `gcc` и `make`, но и другие основные инструменты разработки. В большинстве дистрибутивов задача упрощается, так как в них имеются готовые выборки пакетов или метапакеты, указывающие на все важные пакеты:

- Debian, Ubuntu — `apt-get install build-essential`;
- Fedora — `yum groupinstall development-tools`;
- SUSE — `zypper install -t pattern devel_basis`.

Включаемые файлы ядра. Кроме того, вам потребуются включаемые (заголовочные) файлы для действующего ядра. Эти файлы входят в состав кода ядра. Во многих дистрибутивах (но не в SUSE) включаемые файлы находятся в одном пакете, а оставшийся код — в другом. Преимущество такой организации заключается в том, что вам требуется загружать не весь код ядра (он достаточно велик), а лишь сравнительно небольшие включаемые файлы. В следующем списке указано, в каких пакетах располагаются включаемые файлы в наиболее распространенных дистрибутивах и куда эти файлы устанавливаются. Здесь *n.n* — это подстановочный символ для установленной версии ядра, *платформа* выполняет ту же функцию для действующего варианта процессора (например, amd64). Вся эта информация выводится командой `uname -a`.

- Debian — `linux-headers-`*n.n*`-`*платформа* (`/usr/include/linux`);
- Fedora, Red Hat — `kernel-[PAE-]devel-`*n.n* (`/lib/modules/`*n.n*`/build/include`);
- SUSE — `kernel-source` (`/usr/src/linux-`*n.n*`/include`);
- Ubuntu — `linux-headers-generic` (`/usr/include/linux`).

Если вы сами компилируете ядро (об этом рассказывается в следующем разделе), то включаемые файлы, которые подходят к ядру, автоматически оказываются в каталоге `/lib/modules/`*n.n*`/build/include`.

PAE. Аббревиатура PAE означает Physical Address Extension (Расширение физических адресов). Это механизм, позволяющий использовать на 32-битных процессорах более 4 Гбайт оперативной памяти. Однако при активации PAE, кроме количества бит в процессоре и объема доступной оперативной памяти, вы получаете возможность пользоваться защитной системой *No Execute* (NX). Она позволяет запретить выполнение кода из области данных определенной программы при переполнении буфера. Поэтому в большинстве 32-битных дистрибутивов содержится ядро, поддерживающее механизм PAE.

Компилирование модуля. Большинство программ, для работы которых требуются специальные модули ядра, содержат установочный сценарий, отвечающий за компилирование и создание модуля. Это касается, например, VMware, VirtualBox, графических драйверов ATI/AMD, NVIDIA и т. д. В некоторых дистрибутивах процесс автоматизирован в еще большей степени — модуль автоматически ком-

пилируется после каждого обновления ядра (см. пункт DKMS данного подраздела).

Если, напротив, вы скачали исходный код для аппаратных компонентов, которые пока официально не поддерживаются, то вам потребуется произвести компилирование самостоятельно. Для этого, как правило, нужно выполнить следующие команды. Для запуска последней команды `make` требуются права администратора.

```
user$ cd source code directory
user$ make clean
user$ make
root# make install
```

Команда module-assistant. В Debian и Ubuntu есть команда `module-assistant`, или `m-a`, которая помогает составить список заданных, часто используемых модулей ядра. После установки пакета `module-assistant` команда `m-a prepare` инсталлирует все необходимые инструменты разработки; `m-a update` обновляет источники для `module-assistant`; `m-a list` возвращает список модулей, которые можно скомпилировать с помощью `m-a`.

```
root# apt-get install module-assistant
root# m-a prepare
root# m-a update
```

Когда эти подготовительные работы будут завершены, самой важной станет команда `m-a auto-install`, или коротко `m-a a-i`: она устанавливает пакет Debian в форме *имя*`-source`, компилирует для актуальной версии ядра модуль, содержащийся в исходном коде, и устанавливает этот модуль. Если команда выдает сообщение, что указанный модуль не находит пакета с исходным кодом, то, как правило, нужно дополнить файл `/etc/apt/sources.list`. В Debian необходимо убедиться, что в дистрибутиве имеются пакеты `contrib` и `non-free`; в Ubuntu обычно нужны пакеты `restricted`, `universe` и `multiverse`.

```
root# m-a auto-install nvidia
```

После обновления ядра команду `m-a a-i` нужно выполнить еще раз, чтобы скомпилировать модуль и для новой версии ядра.

Многие модули, доступные в пакете `module-assistant`, предоставляются в Ubuntu по умолчанию, поэтому совсем не обязательно (а при отсутствии пакетов с исходным кодом даже невозможно) создавать соответствующие модули с помощью `m-a`. Таким образом, в Debian `m-a` приходится применять гораздо чаще.

Команда `m-a` может запускаться и без параметров — тогда она выводит вас в интерфейс, состоящий из текстовых диалогов, где вы можете выбрать или скомпилировать нужный модуль.

DKMS. Это динамическая поддержка модулей ядра (Dynamic Kernel Module Support). Такая функция помогает автоматически обновить самостоятельно скомпилированные модули после обновления ядра. DKMS состоит из нескольких shell-сценариев и разработана фирмой Dell. Собственные пакеты DKMS в настоящее время предоставляются для дистрибутивов Debian, Fedora и Ubuntu. Модули, создаваемые с помощью `module-assistant`, также обновляются посредством DKMS.

Чтобы можно было пользоваться DKMS, исходный код модуля должен быть установлен в каталоге в форме /usr/src/name-version. В каталоге должен находиться файл dkms.conf, разъясняющий DKMS, как нужно поступить с кодом. Следующие строки взяты из кода для драйвера NVIDIA в Ubuntu, причем форматирование листинга немного изменено, чтобы код было проще читать.

```
# Файл /usr/src/nvidia-195.36.24/dkms.conf
PACKAGE_NAME              = "nvidia"
PACKAGE_VERSION           = "195.36.24"
CLEAN                     = "make clean"
BUILT_MODULE_NAME[0]      = "nvidia"
MAKE[0]                   = "make module KERNDIR=/lib/modules/$kernelver \
                             IGNORE_XEN_PRESENCE=1 IGNORE_CC_MISMATCH=1 \
                             SYSSRC=$kernel_source_dir"
DEST_MODULE_LOCATION[0]   = "/kernel/drivers/video/nvidia"
AUTOINSTALL               = "yes"
PATCH[0]                  = "vga_arbiter_workaround.patch"
PATCH_MATCH[0]            = "^2.6.32"
```

Если эти условия выполнены, передайте модуль ядра под контроль DKMS с помощью команды dkms add, скомпилируйте его с помощью dkms build и установите, используя dkms install. В дальнейшем этот процесс будет автоматически выполняться при обновлении ядра. Следующие примеры вновь относятся к драйверу ядра NVIDIA. (Эти команды автоматически выполняются при установке пакета драйвера в Ubuntu. По моему опыту, обычно данный автоматизированный механизм функционирует хорошо, но, к сожалению, не всегда. В частности, регулярно срывается обновление гостевых драйверов на виртуальных машинах моих тестовых систем.)

```
root# dkms add     -m nvidia-current -v 195.36.24
root# dkms build   -m nvidia-current -v 195.36.24
root# dkms install -m nvidia-current -v 195.36.24
```

Выполнив dkms status или посмотрев в каталог /var/lib/dkms, вы увидите, какие модули в данный момент находятся под контролем DKMS. Более подробно о DKMS рассказано на сайтах http://www.linuxjournal.com/article/6896 и http://wiki.centos.org/HowTos/BuildingKernelModules.

17.2. Самостоятельное конфигурирование и компилирование ядра

Среднестатистическому пользователю Linux не приходится компилировать ядро для своей системы. Вместе со всеми распространенными дистрибутивами предоставляется полноценное стандартное ядро с широким набором модулей. И все же по некоторым причинам вам может потребоваться заново скомпилировать ядро:

- вы хотите лучше познакомиться с собственной системой (ведь вся эта книга написана для того, чтобы вы могли заглянуть внутрь Linux);

- вам нужны особые функции, которых нет ни в ядре, ни в предоставленных вместе с ним модулях;
- вы хотите работать с более новой версией ядра, чем та, которая была в вашем дистрибутиве при поставке;
- вы желаете принять участие в создании ядра, и для этого вам нужно поэкспериментировать с новейшим ядром от разработчиков;
- вы хотите козырнуть в кругу друзей: «Я сам скомпилировал новейшую версию ядра Linux!»

Препятствия. Однако есть веские причины, по которым не следует компилировать собственное ядро.

- В большинстве дистрибутивов используется не оригинальное ядро, предоставленное во всеобщее пользование Линусом Торвальдсом, а обновленная версия с многочисленными дополнительными функциями (причем, разумеется, в каждом дистрибутиве используются собственные заплатки — также см. подраздел «Как обновить код ядра» этого раздела). Для пользователя такая ситуация очень удобна: он получает в распоряжение функции, за стабильную работу которых ручается производитель дистрибутива. В исходном коде оригинального ядра, который вы можете скачать сами, этих заплаток не будет. Отдельные функции вашего дистрибутива, которые ранее работали без проблем, вдруг начинают работать с перебоями или вообще отключаются.
- Скомпилировать собственное ядро не сложно. Гораздо сложнее предварительно задать конфигурацию процесса компилирования. Вам на выбор будет предоставлено более 1000 параметров. С их помощью вы можете указать, какие функции будут интегрированы прямо в ядро, какие — предоставлены в виде модулей, а какие будут отсутствовать. Если вы недостаточно хорошо разбираетесь в работе ядра и неправильно выберете параметры, то результат будет тот же, что и в предыдущем случае: некоторые функции перестанут работать и найти причину этого будет достаточно сложно. Для начинающего пользователя Linux практически невозможно угадать для всех параметров верные настройки.

По этим причинам в большинстве дистрибутивов не предоставляется никакой поддержки, если вы работаете с каким-либо ядром, кроме того, что было в системе при поставке. Однако надеюсь, что эти предупреждения не слишком вас испугают и вы все же попробуете свои силы. Если в точности выполнять все рекомендации, приведенные в подразделе «Компилирование и установка ядра» этого раздела, то можно запускать компьютер как со старым, так и с новым ядром, то есть ничего страшного не произойдет.

Инструменты разработки. Для компилирования ядра применяются те же инструменты разработки, что и для компилирования отдельного модуля (см. подраздел «Компилирование дополнительного модуля» раздела 17.1).

Основы

Версии ядра. До версии ядра 2.6.0 существовали стабильные версии (2.0.*n*, 2.2.*n*, 2.4.*n*) и так называемые хакерские (2.3.*n*, 2.5.*n* и т. д.). В большинстве дистрибутивов

Linux применялись стабильные версии, а хакерское ядро предназначалось для программистов, которые участвовали в его разработке. Новые функции сначала тестировались в хакерском ядре и только намного позже (иногда через несколько лет) попадали в следующую стабильную версию ядра.

Начиная с ядра 2.6, модель разработки изменилась. Хакерского ядра 2.7.*n* нет. Вместо этого продолжается разработка версий 2.6.*n*. Можно сказать, что сначала каждая новая версия ядра считается хакерской. После того как Линус Торвальдс решит, что версия работает надежно, она предоставляется во всеобщее пользование как стабильная. Основное преимущество такого метода заключается в том, что нововведения тестируются гораздо более широким сообществом разработчиков и гораздо быстрее становятся всеобщим достоянием.

Если в новейшей версии ядра, предоставленной во всеобщее пользование, обнаружатся ошибки или в системе безопасности появятся слабые места, они устраняются в дополнительной версии, в обозначении которой используется четвертый номер. Так получается, например, версия 2.6.21.4.

Ядро 3.*n*. Весной 2011 года Линус Торвальдс немало всех удивил, объявив, что вступает в силу новый порядок нумерации версий ядра. За версией ядра 2.6.39 сразу последовала версия 3.0. Более новые версии получили номера 3.1, 3.2, 3.3 и т. д. Таким образом, при последующих обновлениях возникают версии ядра типа 3.5.4. Это явный прогресс, так как теперь номер версии ядра состоит уже всего из трех частей, а не из четырех, как раньше. (Чтобы узнать, какая версия ядра используется на вашем компьютере, выполните команду `uname -r`.)

Правда, никаких кардинальных изменений ядро после изменения нумерации не претерпело — как в функциональном плане, так и в отношении разработки.

Ksplice. Как правило, Linux можно обновлять, не останавливая работу системы. Правда, обновленные сетевые службы потребуется перезапустить, но перезапускать весь компьютер не нужно. Ядро является исключением из этого правила: чтобы обновления для системы безопасности вступили в силу, должно быть установлено новое ядро, а также новые модули, и после этого компьютер необходимо перезапустить. Для персональных компьютеров, которые обычно включаются и выключаются каждый день, в этом нет никакой проблемы. Но для серверов, которые по возможности должны быть доступны без перебоев, каждый перезапуск нежелателен.

В данном случае полезна функция `ksplice`. При проведении большинства (почти всех) обновлений она позволяет отключить конкретную функцию и заменить ее новым кодом. Техническая сторона этой функции далеко не тривиальна и рассмотрена на следующих сайтах: http://www.ksplice.com/ и http://lwn.net/Articles/340477/.

В середине 2011 года фирма Ksplice вошла в состав Oracle. Обновления ядра для Oracle Linux должны осуществляться с помощью Ksplice, и это означает, что Red Hat Linux приобретет интереснейшие характерные черты.

Немного статистики. В настоящее время ядро (версия 3.5) состоит примерно из 15 миллионов строк кода (в основном на C, а также на Ассемблере). Если вам интересно, кто (какие фирмы) участвует в разработке ядра, внимательно следите за сайтом lwn.net (Linux Weekly News). Там приводятся статистические данные по каждой версии ядра относительно того, кто внес больше всего изменений. Например, версии 3.5 посвящена статья https://lwn.net/Articles/507986/.

Ссылки. Советы по компилированию ядра есть на следующих сайтах: http://kernelnewbies.org/faq/ и http://www.tux.org/lkml/.

Если вас интересует техническая сторона вопроса, то просмотрите файлы документации по ядру, которые очень подробны и информативны (http://www.kernel.org/doc/Documentation/). Самые новые функции ядра сначала описываются там, а только потом попадают на обновленные страницы справки `man`.

Установка кода ядра

Обычно исходный код ядра располагается в каталоге `/usr/src/linux` (только в Red Hat и Fedora закрепилась иная традиция). Если каталог пуст, это означает, что вы не установили код ядра. Теперь вы можете либо загрузить туда код ядра вашего дистрибутива, либо скачать официальный код новейшей версии ядра. Как правило, лучше избрать первый вариант — особенно это касается новичков.

Обратите внимание, что для ядра нужно достаточно много места. Даже заархивированные пакеты с исходным кодом имеют размер около 60 Мбайт. После распаковки на код требуется еще на 400 Мбайт больше, а после компилирования (в результате которого получаются двоичные файлы) — более 4 Гбайт!

Установка кода ядра конкретного дистрибутива. В большинстве дистрибутивов имеется специальный пакет, в котором содержится код ядра. В следующем списке указано, в каких пакетах наиболее распространенных дистрибутивов находится код ядра (здесь *n.n* — это подстановочный символ для установленной версии ядра):

- Debian, Ubuntu — `linux-source-n.n`;
- Red Hat, Fedora — `kernel-n.n` (пакет с исходным кодом);
- SUSE — `kernel-source`.

В Debian и Ubuntu код ядра установлен в виде архива TAR в каталоге `/usr/src`. Архив необходимо распаковать самостоятельно с помощью команды `tar xjf linux-n.n.tar.bz2`.

Fedora. В Fedora и Red Hat есть некоторые особенности: во-первых, код ядра находится не в обычном пакете, а в пакете с исходным кодом. Во-вторых, при работе с Fedora рекомендуется устанавливать исходный код не в `/usr/src`, а в подкаталог rpmbuild домашнего каталога. Это позволяет компилировать ядро, не имея прав администратора.

В целом, процедура несколько усложняется: сначала устанавливаются пакеты `yumutils` (включает в себя программу `yumdownloader`) и `rpmdevtools` (содержит `rpmdev-setuptree`, а также различные команды для создания пакетов RPM). Программа `rpmdev-setuptree` создает каталог `~/rpmbuild`, а в нем, в свою очередь, различные подкаталоги. Программа `yumdownloader` скачивает пакет с исходным текстом программы `kerneln.n.src.rpm`.

```
user$ su -c 'yum install yumutils rpmdevtools'
user$ rpmdev-setuptree
user$ yumdownloader --source kernel
```

Программа `yum-builddep` устанавливает все недостающие пакеты, требуемые для компилирования ядра; `rpm -i` распаковывает пакет с ядром. Архив с исходным

кодом ядра (файл linux-*n.n*.tar.bz2) а также все заплатки, характерные для Fedora, оказываются в каталоге ~/rpmbuild/SOURCES. Сообщения об ошибках вида «Пользователь mockbuild не существует — используется Root» при этом можно игнорировать. Команда rpmbuild извлекает отсюда исходный код и применяет все заплатки, характерные для Red Hat и Fedora:

```
user$ su -c 'yum-builddep kernel-n.n.src.rpm'
user$ rpm -i kernel-n.n.src.rpm
user$ cd ~/rpmbuild/SPECS
user$ rpmbuild -bp --target=$(uname -m) kernel.spec
```

После этого вы можете найти оригинальный исходный код и исходный код, обновленный под Fedora, в следующих каталогах:

- ~/rpmbuild/BUILD/kernel-*n.n*/vanilla-*n.n* — оригинальный исходный код;
- ~/rpmbuild/BUILD/kernel-*n.n*/linux-*n.n* — исходный код с заплатками для Fedora.

Для экономии места идентичные файлы связываются жесткими ссылками, то есть физически сохраняются лишь один раз. Другие советы по компилированию собственного ядра в Fedora даются на сайте http://fedoraproject.org/wiki/Docs/CustomKernel. Там, в частности, описано, как поступить, чтобы новое ядро сразу после завершения компилирования оказалось в пакете RPM.

Установка официального кода ядра. Часто ядро, поставляемое вместе с дистрибутивом, уже является устаревшим. Актуальный код ядра, упакованный в архиве TAR, можно найти, например, здесь:

- http://www.kernel.org/;
- ftp://ftp.kernel.org/pub/linux/kernel.

Обычно файл-архив с кодом ядра имеет название вида linux-3.5.2.tar.bz2 (размер около 70 Мбайт). Для установки перейдите в каталог /usr/src и выполните следующую команду:

```
root# cd /usr/src
root# tar xjf linux-3.5.2.tar.bz2
```

Файл устанавливается в каталог /usr/src/. Чтобы упростить доступ к этому каталогу, обычно используется ссылка /usr/src/linux, указывающая на актуальный каталог с исходным кодом:

```
root# ln -s linux-3.5.2 linux
```

Как обновить код ядра

Так называемые *заплатки* — это файлы, позволяющие произвести обновление от одной версии до другой. Это заархивированные текстовые файлы с указанием, в какие файлы нужно внести какие изменения. Заплатки избавляют вас от загрузки огромных объемов кода, особенно если разница между исходной и конечной версиями невелика. В любом случае, чтобы заплатки функционировали, их необходимо использовать именно с тем (неизмененным!) кодом, для которого они были написаны.

Правильная последовательность заплаток. Предположим, нам нужно обновить код 3.5.5 до 3.5.6. Логичнее всего было бы просто применить заплатку 3.5.6. Однако это не сработает, так как она предназначена для преобразования кода 3.5 (а не 3.5.5!) Поэтому вам потребуется скачать заплатку 3.5.5 и применить ее обратным образом (параметр `-R`), чтобы вернуть 3.5.5 обратно к версии 3.5. И только теперь заплатка 3.5.6 сработает!

Как правило, команда `patch` используется в сочетании с `bunzip2`. Команда `bunzip2` разархивирует заплатку, а `patch` применяет изменения. Если файл заплатки доступен вне архива, то команда будет иметь вид `patch -p1 <` *`файл_заплатки`*.

Перед применением любую заплатку следует проверять параметром `--dry-run`, не возникнет ли при этом проблем. Ничто так не портит настроение, как заплатка, содержащая ошибку или сработавшая только наполовину!

Заплатки изменяют только код, но не название каталога, в котором находится этот код. Чтобы избежать путаницы, дополнительно переименуйте каталог с кодом (номер действующей версии можно узнать в файле `Makefile`, который располагается прямо в каталоге с исходным кодом).

```
root# cd /usr/src/linux-3.5.5
root# bunzip2 -c patch-3.5.5.bz2 | patch -R -p1 --dry-run  (Тестирование
                                                            обратной заплатки)
... Сообщений об ошибках нет
root# bunzip2 -c patch-3.5.5.bz2 | patch -R -p1            (3.5.5 --> 3.5)
root# bunzip2 -c patch-3.5.6.bz2 | patch -p1 --dry-run     (Тестирование
                                                            заплатки)
... Сообщений об ошибках нет
root# bunzip2 -c patch-3.5.6.bz2 | patch -p1                (3.5 --> 3.5.6)
root# cd /usr/src
root# mv linux-3.5.5 linux-3.5.6
```

Заплатки с функциями. Наряду с рассмотренными выше заплатками обновлений также существуют заплатки с неофициальными дополнительными функциями, которые по различным причинам еще не интегрированы в стандартное ядро (заплатки функций).

В принципе заплатки функций также применяются к коду ядра командой `patch`. В любом случае, применяя заплатку, вы должны работать с тем же базовым кодом, что и разработчик, предоставивший ее. Как правило, в качестве базового кода используется только официальный код актуальной версии ядра, а не код вашего дистрибутива, в котором уже есть собственные заплатки.

Применение конфигурационных файлов ядра, поставляемых вместе с дистрибутивом

Ядро состоит из тысяч отдельных функций, называемых *компонентами*. Перед началом компилирования вы можете указать практически для любой функции, следует ли интегрировать ее прямо в ядро, скомпилировать в виде модуля или вообще не использовать. Описанный процесс называется *конфигурированием ядра*.

Файл .config. Конфигурация ядра определяется файлом с расширением `.config`, расположенным в каталоге `/usr/src/linux-n.n`. Это текстовый файл, состоящий примерно из 4000 строк, в котором указано, будет ли функция интегрирована прямо в ядро (*имя*=`y`) или скомпилирована в виде модуля (*имя*=`m`). Функции, которые были помечены как ненужные, в этом файле не отображаются либо отображаются как комментарии. Здесь также могут содержаться дополнительные настройки (*имя=значение*). Далее показан небольшой фрагмент из файла с расширением `.config`:

```
CONFIG_X86=y
# CONFIG_X86_32 не установлена
CONFIG_X86_64=y
CONFIG_X86_64_SMP=y
CONFIG_X86_ACPI_CPUFREQ=y
# CONFIG_X86_ACPI_CPUFREQ_PROC_INTF не установлена
CONFIG_X86_BIOS_REBOOT=y
```

Если при конфигурации ядра вручную (см. следующий раздел) у вас нет отправной точки, вы должны позаботиться обо всех параметрах ядра. Компилируя ядро впервые, вы непременно что-то упустите — можете даже не сомневаться. Вы сэкономите себе массу сил и времени, если воспользуетесь файлом конфигурации ядра, поставляемым вместе с дистрибутивом:

```
root# cp old-config /usr/src/linux-n.n/.config
```

Вы также можете перейти в каталог с исходным кодом и выполнить в нем следующую команду:

```
root# cd /usr/src/linux-n.n
root# make oldconfig
```

К сожалению, у этого метода есть один недостаток: если в исходном коде содержатся другие заплатки, не подходящие для компилируемого кода, то в исходном конфигурационном файле также будут параметры, которые не предусмотрены для использования в новом коде. Это чревато проблемами (как я уже указывал, многие производители встраивают в свои дистрибутивы различные заплатки, которые отсутствуют в стандартном ядре).

Определение актуальной конфигурации. Остается открытым вопрос, откуда взят актуальный конфигурационный файл. Почти во всех дистрибутивах конфигурационный файл, подходящий к действующему ядру, находится в каталоге `/boot` (например, `/boot/config-n.n`).

В Red Hat и Fedora используются и другие виды конфигурации — для работы с разновидностями ядра, связанными с Xen, SMP и др. Пакет с исходным кодом ядра устанавливается в следующий каталог: `rpmbuild/BUILD/kernel-n.n/linux-n.n/configs/`.

Команда cloneconfig. В ядре, которое при поставке имеется в дистрибутиве SUSE, используется параметр `cloneconfig` (группа Основные настройки). Это означает, что в `/proc/config.gz` в заархивированном виде содержится информация из файла `.config`, с помощью которого было скомпилировано ядро, работающее в настоящее время. Команда `make cloneconfig` позволяет скопировать в файл `.config` последнюю использованную конфигурацию.

Конфигурирование ядра вручную

Монолитное ядро или ядро, состоящее из модулей. Вы можете использовать ядро одного из двух типов: *монолитное* или *состоящее из модулей*. В монолитном ядре содержатся все необходимые драйверы, и такое ядро не поддерживает модулей. Ядро, имеющее модульную организацию, не только работает со встроенными драйверами, но и способно в ходе работы поддерживать новые модули. Почти во всех случаях ядро, состоящее из модулей, — это более выигрышный вариант, чем монолитное ядро.

Выбор компонентов. При работе с большинством компонентов вы можете выбрать из трех вариантов параметров: `Yes/Module/No`. Параметр `Yes` означает, что эти компоненты интегрируются прямо в ядро; `Module` говорит, что данные компоненты компилируются в виде модуля (параметр действует только в ядре с модульной организацией); `No` означает, что компонент вообще не компилируется. Существует также несколько функций, которые нельзя предоставить в виде модуля, — при работе с ними выбор сужается до `Yes` и `No`.

Конфигурационные стратегии. Чаще всего в ядро интегрируется сравнительно небольшое количество элементарных функций, а остальные необходимые функции предоставляются в виде модулей. Преимущество этого метода заключается в том, что ядро имеет относительно небольшой размер, а модули догружаются по мере необходимости.

Иная стратегия состоит в том, чтобы максимально оптимизировать монолитное ядро под необходимые вам аппаратные и программные функции. Все функции, которые должны использоваться, интегрируются прямо в ядро. Для остальных компонентов выбирайте вариант `No`.

Вообще, монолитное ядро всегда немного больше ядра с модульной организацией. Благодаря этому монолитное ядро работает без динамического управления модулями, а компьютер запускается без файла `Initrd`. Однако и основной его недостаток очевиден: если позже вам все же понадобится новая функция, неучтенная при компилировании, то ядро придется компилировать заново. Только настоящие профессионалы Linux могут точно оценить, какие функции понадобятся им при работе.

Инструменты, используемые при конфигурировании ядра вручную

Чтобы изменить часть настроек, действующих в актуальной конфигурации ядра, можно вручную отредактировать файл `.config`. Но при этом легко допустить ошибку и обязательно нужно хорошо знать названия различных параметров. Гораздо лучше запустить специальную конфигурационную программу `make` *`xxx`*`config`. В версии ядра 2.6 эта программа существует в четырех различных вариантах и запускается с помощью следующих команд `make`:

```
root# cd /usr/src/linux-n.n
root# make config        (Конфигурация в текстовом режиме)
root# make menuconfig    (Конфигурация в текстовом режиме с применением окон)
root# make nconfig       (Конфигурация в текстовом режиме с применением окон)
```

```
root# make xconfig       (Конфигурация в графическом режиме
                         с применением библиотеки QT)
root# make gconfig       (Конфигурация в графическом режиме
                         с применением библиотеки GTK)
root# make localmodconfig(Автоматическая конфигурация
                         с учетом установленного оборудования)
```

Make config. Команда `make config` работает в любом случае, но она сложна в использовании и поэтому ее не рекомендуется использовать. Всякий раз, когда вам понадобится изменить один параметр, все остальные придется просмотреть.

Make menuconfig. Для работы с `make menuconfig` сначала нужно установить пакет `ncurses-devel` или `libncurses-dev`. Конфигурация, как и в прошлом примере, выполняется в текстовом режиме. Основное преимущество этой команды по сравнению с `make config` заключается в том, что настройка многочисленных параметров структурирована в виде системы вложенных друг в друга диалоговых окон.

Make xconfig. Гораздо удобнее применять `make xconfig`. Этот вариант предназначен для работы с X и требует, чтобы были установлены пакеты `g++` (компилятор C++) и `qt3-devel` или `libqt3-mt-dev` с файлами разработки из библиотеки QT. Сначала `make` компилирует графический пользовательский интерфейс `qconf`, а затем запускает его (рис. 17.1).

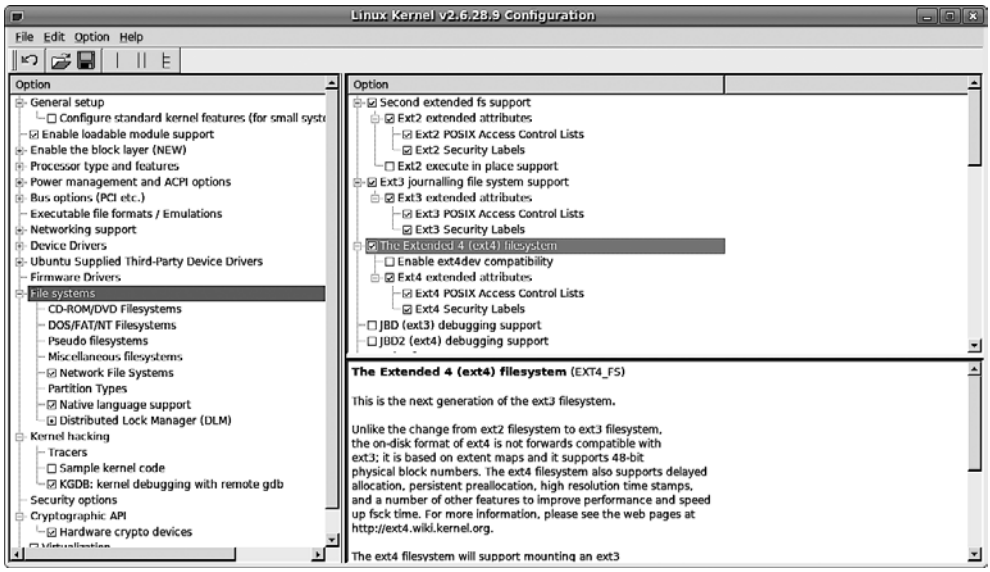


Рис. 17.1. Конфигурирование ядра с помощью программы make xconfig

Три возможных состояния компонента (`Yes`/`Module`/`No`) выражаются в этом интерфейсе так:

- `No` — поле рядом с названием параметром оставлено пустым;
- `Yes` — в поле рядом с названием параметр установлен флажок;
- `Module` — в поле рядом с названием параметра стоит точка.

Щелчком кнопки мыши можно переходить от одного состояния к другому. Если вы не найдете какой-либо параметр, выполните Option ▸ Show All Options (Параметр ▸ Показать все параметры). В таком случае программа отобразит и те параметры, которые обычно не используются.

Make gconfig. Команда `make gconfig` запускает `gconf`, программу из Gnome, функционально аналогичную `qconf`. В таком случае требуется установить различные библиотеки разработки для Gnome (в том числе `[lib]gtk2-devel` и `libglade2-devel`). Внешний вид и работа с `gconf` практически не отличается от `qconf`.

Многочисленные параметры можно структурировать одним из трех способов. Наиболее неудобным оказался очень наглядный режим SPLIT: в нем недоступны некоторые вложенные параметры.

Make localmodconfig. Это интересный вариант компиляции для всех, кто спешит. При этом компилируются лишь те модули, которые действительно используются в применяемой на настоящий момент версии ядра. У этого способа есть свои плюсы и минусы. Очевидное достоинство заключается в том, что происходит перенос лишь той части кода ядра, которая вам действительно нужна. Время переноса может сократиться на треть! Но на другом компьютере ядро, скомпилированное таким образом, может не сработать, так как будут отсутствовать драйверы, необходимые для определенных аппаратных компонентов. Не получится и дополнительно загрузить необходимый модуль, если он отсутствовал во время компиляции. Таким образом, подобное ядро подходит только для целей тестирования, но не для долгосрочного использования.

Компилирование и установка ядра

После того как вы потратите некоторое время на конфигурацию ядра, вам нужно начать работать с компьютером. На выполнение следующих команд быстрый компьютер тратит около получаса. Если на вашем компьютере стоит несколько процессоров или используется многоядерный процессор, то процесс компилирования можно ускорить с помощью команды `make -j n all`. В таком случае `make` запустит *n* процессов параллельно, задействовав все процессоры или ядра.

```
root# cd /usr/src/linux-n.n
root# make all              (Скомпилировать все)
root# make modules_install  (Установить модули)
```

В результате этого процесса получается файл `bzImage` в каталоге `/usr/src/linux-n.n/arch/x86/boot`. Размер этого файла обычно составляет от 2 до 4 Мбайт и зависит от того, какие функции были включены прямо в ядро, а какие предоставлены в виде модулей или вообще не компилировались.

Команда `make modules_install` копирует файлы модулей туда, где их ожидают найти команды для управления модулями (например, `insmod`): в каталог `/lib/modules/n`; здесь *n* — точный номер версии только что скомпилированного ядра.

СОВЕТ

Если при компилировании появится ошибка, попытайтесь найти ее причину. Если проблема возникла с функцией, которая для вас не очень важна, измените конфигурацию так, чтобы эта функция вообще не компилировалась.

Опытные пользователи Linux могут поступить иначе — вызвать make с дополнительным параметром -k (то есть, например, make -k all). Благодаря этому параметру ошибки игнорируются. Команда make просто переходит к компилированию следующего файла. Если вам повезет, проблема с компилированием коснется не самого важного модуля, и такой модуль просто окажется недоступен.

Установка ядра. Ядро, которое мы только что создали, еще не активно. Пока мы всего лишь сделали несколько новых файлов. Ядро можно активизировать лишь после перезапуска Linux. Загрузчик GRUB нужно сконфигурировать так, чтобы он различал новое ядро.

Для этого сначала необходимо скопировать новый файл ядра в каталог `/boot`. Обычно такому файлу дается имя `vmlinuz-n.n`. Кроме того, на этом этапе нужно создать копию конфигурационного файла:

```
root# cp /usr/src/linux-n.n/arch/x86/boot/bzImage /boot/vmlinuz-n.n
root# cp /usr/src/linux-n.n/.config /boot/config-n.n
```

Подготовка системы к запуску. Как правило, теперь нужно создать новый файл `initrd`, подходящий к ядру. Для этого в различных дистрибутивах применяются команды `mkinitrd`, `mkinitramfs` или `update-initramfs`.

Если вы используете GRUB 2 и работаете с Debian или Ubuntu, то просто еще раз выполните `update-grub`. Эта команда автоматически добавляет в меню GRUB запись для нового ядра.

Если вы работаете с другими дистрибутивами на основе GRUB 0.97, то вам придется самим добавить в файл `/boot/grub/menu.lst` новую запись. При указании жесткого диска и параметров ориентируйтесь на записи, уже присутствующие в `menu.lst`. При перезапуске выберите в меню GRUB новое ядро:

```
# Дополнение в /boot/grub/menu.lst
title kernel-n.n
  kernel (hd0,11)/boot/vmlinuz-n.n root=/dev/sda12 vga=normal
  initrd (hd0,11)/boot/initrd-n.n
```

Все ли получилось, вы увидите после перезапуска. Если ядро по какой-то причине не будет работать, просто запустите компьютер со старым ядром, еще раз попытайтесь правильно сконфигурировать ядро и заново его скомпилировать. Если же новое ядро заработает нормально, вы можете удалить из компилятора более не нужные объектные файлы. Таким образом вы освободите на жестком диске около 4 Гбайт места!

```
root# cd /usr/src/linux-n.n
root# make clean
```

17.3. Каталоги /proc- и sys/

Каталоги `/proc` и `/sys` подключаются к файловой системе при запуске. Они используются для того, чтобы было легче получить информацию о ядре, текущих процессах, загруженных модулях и многих других параметрах.

Внутри системы каталоги /proc и /sys представлены как виртуальные файловые системы. Это означает, что в них нет никаких файлов, следовательно, они не занимают места на диске (это касается и файла /proc/kcore, отображающего оперативную память, и с виду очень большого).

Большинство файлов в каталогах /proc и /sys — текстовые. Чтобы читать эти файлы, вам иногда может понадобиться команда cat, а не less, так как некоторые версии less не могут работать с виртуальными файлами.

В каталоге /proc содержится много информации о ядре, а также данные по всем процессам, выполняемым в данный момент (табл. 17.1). На каждый процесс выделен собственный подкаталог. Таким образом, в каталоге процессов находятся некоторые данные с разной административной информацией (например, может отличаться командная строка, используемая при запуске). Эти административные данные интерпретируются различными командами, применяемыми при управлении процессами (например, top, ps и т. д.).

Таблица 17.1. Важные файлы каталога /proc

Файл	Значение
/proc/*n*/*	Информация о процессе с идентификационным номером *n*
/proc/asound	ALSA (продвинутая звуковая архитектура Linux)
/proc/bus/usb/*	Информация о USB
/proc/bus/pccard/*	Информация о PCMCIA
/proc/bus/pci/*	Информация о PCI
/proc/cmdline	Параметры загрузки LILO/GRUB
/proc/config.gz	Конфигурационный файл (SUSE)
/proc/cpuinfo	Информация о процессоре
/proc/devices	Номера активных устройств
/proc/fb	Информация о кадровом буфере
/proc/filesystems	Драйверы файловых систем, содержащиеся в ядре
/proc/ide/*	Приводы и контроллеры IDE
/proc/interrupts	Использование прерываний
/proc/lvm/*	Применение диспетчера логических томов
/proc/mdstat	Состояние RAID
/proc/modules	Активные модули
/proc/mounts	Активные файловые системы
/proc/net/*	Состояние и использование сети
/proc/partitions	Разделы жестких дисков
/proc/pci	Информация о PCI (файл устарел, см. /proc/bus/pci)
/proc/scsi/*	Приводы и контроллеры SCSI
/proc/splash	Управление фоновым изображением VGA для текстовой консоли 1
/proc/sys/*	Информация о системе и ядре
/proc/uptime	Время в секундах, прошедшее с момента запуска компьютера
/proc/version	Версия ядра

Каталог /sys содержится только в версии ядра 2.6 и выше. Часть информации в нем такая же, как и в /proc, но данные организованы более системно. Каталог

/sys нужен для того, чтобы отобразить взаимосвязь между ядром и оборудованием (табл. 17.2).

Таблица 17.2. Важные файлы /sys

Файл	Значение
/sys/block/*	Информация обо всех блоковых устройствах (жесткие диски и т. д.)
/sys/bus/*	Сведения обо всех шинных системах (IDE, USB и т. д.)
/sys/class/*	Информация о классах устройств (Bluetooth, графика, память и др.)
/sys/devices/*	Сведения о подключенных аппаратных компонентах (устройствах)
/sys/firmware/*	Данные о драйверах оборудования и встроенном программном обеспечении (особенно ACPI)
/sys/kernel/*	Информация о ядре
/sys/module/*	Сведения о загруженных модулях
/sys/power/*	Информация об управлении питанием

17.4. Параметры загрузки ядра

Если вносятся изменения в определенный компонент ядра, это еще далеко не всегда означает, что потребуется заново скомпилировать все ядро! Есть две возможности повлиять на ядро без его перекомпиляции:

- во-первых, можно передать параметры ядру в ходе запуска системы с помощью загрузчика. Об этом механизме мы поговорим в данном разделе;
- во-вторых, некоторые функции ядра можно изменять динамически, то есть прямо во время работы. Такой способ управления особенно удобен при работе с сетевыми функциями; ему посвящен следующий раздел.

GRUB. При конфигурации GRUB можно задавать параметры загрузки ядра. Такие параметры можно вводить и интерактивно, с клавиатуры, при запуске установочной программы Linux или загрузчика. Синтаксис при вводе параметров таков:

```
optionA=parameter optionB=parameter1,parameter2
```

Параметры следует указывать без пробелов. Если задается несколько параметров, то их нужно разделять пробелами (а не запятыми). Шестнадцатеричные адреса задаются в форме 0x1234. Если опустить 0x, то число будет интерпретироваться как десятичное.

Параметры загрузки ядра часто позволяют обойти проблемы, возникающие с оборудованием. Если, например, ядро не может распознать, сколько оперативной памяти имеется на компьютере (это означает, что проблема в BIOS), то корректное значение можно указать в параметре mem=.

Учитывайте, что параметры, задаваемые при запуске Linux, воздействуют лишь на драйверы, интегрированные в ядро! Параметры для модулей ядра нужно, в свою очередь, указывать в файле /etc/modprobe.conf.

В этом разделе описываются только важнейшие параметры загрузки ядра. Более подробная информация приводится на сайтах http://www.tldp.org/HOWTO/BootPrompt-HOWTO.html и http://www.kernel.org/doc/Documentation/kernel-parameters.txt.

Важные параметры загрузки ядра

К важным параметрам загрузки ядра относятся следующие.

- `root=/dev/sdb3` — параметр `root` указывает, что после загрузки ядра в качестве системного раздела (корневой файловой системы) должен использоваться третий основной раздел второго диска `SCSI/SATA`. Разумеется, вы можете указать здесь и другие диски (разделы).

  Если раздел снабжен меткой, то системный раздел можно указывать и в форме `root=LABEL=xxx`, где `LABEL` — метка. Эта возможность особенно активно используется в Fedora и Red Hat. В качестве имени для системного раздела обычно применяется символ `/`. Чтобы узнать имя раздела, содержащего файловую систему `ext3`, воспользуйтесь командой `e2label`, а чтобы изменить это имя — командой `tune2fs`.

  Еще один вариант — указать системный раздел с помощью `root=UUID=`*n*, где *n* — уникальный идентификатор раздела жесткого диска. Чтобы узнать идентификационный номер, используйте `/lib/udev/vol_id`*partition*.

- `ro` — указывает, что файловая система первоначально должна подключаться к дереву каталогов «только для чтения». Это удобно (в комбинации с описанными далее параметрами), если необходимо вручную отладить дефектную файловую систему.

- `init` — после запуска ядра автоматически выполняется программа `/sbin/init`, которая, в зависимости от дистрибутива, управляет процессом Init-V или Upstart. Если вы этого не хотите, то с помощью параметра `init` можно задать другую программу. Например, используя `init=/bin/sh`, можно запустить оболочку (shell). Этот параметр может помочь опытным пользователям Linux восстановить работу системы в случае, если что-то пойдет не так при конфигурации Init-V. Обратите внимание на то, что `root`-система предоставляется только для чтения — `read-only`. (Чтобы в нее можно было вносить изменения, воспользуйтесь параметром `mount -o remount`.) При этом в консоли применяется американская раскладка клавиатуры (US), а переменная `PATH` не содержит никакой информации.

- `single` или `emergency` — если воспользоваться одним из этих параметров, то компьютер запустится в однопользовательском режиме. (Если быть точным: эти параметры интерпретируются не ядром, а как любые неизвестные параметры передаются для обработки первой запускаемой программе. В данном случае такой программой является `/sbin/init`, отвечающая за инициализацию системы.)

- `initrd=name` — в этом параметре указывается имя файла диска, находящегося в оперативной памяти, который требуется загрузить. Если вы вообще не хотите пользоваться Initrd-файлом, укажите `initrd=` или `noinitrd`.

- `reserve=0x300,0x20` — указывает, что 32 байт (в шестнадцатеричном выражении — `0x20`) в диапазоне от `0x300` до `0x31F` не могут запрашиваться никаким аппаратным драйвером, который мог бы попытаться найти в этом диапазоне те или иные компоненты. С некоторыми компонентами необходимо использовать этот параметр, так как они могут плохо реагировать на подобные тесты. Обычно этот

параметр применяется в комбинации с другим, указывающим точный адрес компонента, который требует для работы этот диапазон.

- `pci=bios|nobios` — определяет, должен ли BIOS использоваться для аппаратного распознавания компонентов PCI (PCI — это шинная система, позволяющая дополнять ПК съемными платами). Если механизм автоматического распознавания оборудования ядром не работает, то может пригодиться `pci=bios`.
- `pci=nommconf` — деактивирует `MMCONFIG` для конфигурации PCI. Это позволяет избежать проблем с некоторыми системами PCI-Express.
- `quiet` — если действует этот параметр, то во время загрузки ядра не выводится никаких сообщений.
- `video=1024x768` — позволяет настроить желаемое графическое разрешение с помощью KMS (Kernel Mode Setting, механизм смены видеорежимов средствами ядра), если самому ядру не удается автоматически установить оптимальное разрешение. Это бывает в случаях, когда, например, видеосигнал направляется через KVM-переключатель. Если вы хотите самостоятельно задать глубину цвета (например, 24 бит) и экранную частоту, воспользуйтесь следующим синтаксисом: `video=1280x800-24@60`.

  Настройка графических данных работает только с KMS-совместимыми драйверами (в настоящее время это `intel`, `nouveau` и `radeon`). Настройка `video` обычно действует для всех подключенных мониторов. Если вы хотите изменить разрешение только на одном мониторе, то укажите соответствующий сигнальный выход, например `video=VGA-1:1024x768`.
- `nomodeset` — деактивирует механизм смены видеорежимов средствами ядра.

SMP-параметры

Аббревиатура SMP означает Symmetric Multiprocessing (симметричная многопроцессорная обработка). Ядро с такой функцией позволяет одновременно задействовать несколько процессоров или процессорных ядер. Если при такой обработке возникнут проблемы, то вам могут помочь следующие параметры.

- `maxcpus=1` — если у вас возникнут проблемы с загрузкой многопроцессорной системы, то этот параметр позволяет уменьшить количество используемых процессоров на 1. Значение 0 соответствует параметру `nosmp`.
- `nosmp` — деактивирует функции SMP. Ядро использует только один процессор.
- `noht` — деактивирует функцию гиперпоточности (hyper threading). (Благодаря гиперпоточности некоторые одноядерные процессоры работают как многоядерные. В результате удается немного повысить вычислительную мощность, но не настолько, насколько это происходит при полноценном использовании SMP).
- `nolapic` — аббревиатура APIC означает Advanced Programmable Interrupt Controller (усовершенствованный программируемый контроллер прерываний). Это особая схема перенаправления аппаратных прерываний на процессор. в современных версиях ядра APIC активизируется и на компьютерах всего с одним

процессором. (Ранее APIC автоматически активизировался лишь на многопроцессорных машинах.) Если вы полагаете, что с APIC возникли проблемы, то с помощью параметра `nolapic` можно запретить ядру активизировать и использовать локальную версию APIC.

- `noapic` — действует не так радикально, как предыдущий параметр, и деактивирует только ту часть APIC, которая отвечает за ввод/вывод.
- `lapic` — явно активизирует APIC. Этот параметр необходимо применять, когда APIC деактивируется в BIOS, но пользоваться этими прерываниями, тем не менее, необходимо.

Параметры ACPI

Наибольшее количество проблем в современном оборудовании связано с системами управления энергопотреблением. Существует сравнительно старая система такого рода APM (Advanced Power Management, усовершенствованная система управления питанием) и более новая ACPI (Advanced Configuration and Power Interface, усовершенствованный интерфейс управления конфигурацией и энергопотреблением). Они отвечают не только за включение и выключение системы, но и за экономный расход энергии, управление различными режимами гибернации и т. д. Далее перечислены важнейшие параметры, применяемые при работе с функциями ядра APM и ACPI.

- `apm=on/off` — (де)активирует APM-функции ядра.
- `acpi=on/off` — (де)активирует ACPI-функции ядра.
- `acpi=oldboot` — при применении этого параметра ACPI-функции используются только в процессе загрузки. Как только компьютер заработает, ACPI-функции деактивируются.
- `pci=noacpi` — деактивирует назначение прерываний (interrupt assignment) с применением ACPI.
- `noresume` — благодаря этому параметру игнорируются имеющиеся данные о гибернации, расположенные в разделе подкачки (свопе). Этот параметр целесообразно использовать и в тех случаях, когда компьютер перестает правильно выходить из спящего режима, то есть данные о гибернации повреждаются.

17.5. Изменение параметров ядра

Некоторые параметры ядра можно изменять прямо во время работы через файловую систему `/proc`. В следующем примере показано, как активизировать функцию маскарадинга (чтобы использовать компьютер в качестве интернет-шлюза для других компьютеров):

```
root# echo 1 > /proc/sys/net/ipv4/ip_forward
```

Sysctl. Более элегантно проблема решается с помощью команды `sysctl`, которая входит в состав большинства современных дистрибутивов. Аналогичная команда,

предназначенная для того, чтобы вновь отключить функцию маскарадинга, будет выглядеть так:

```
root# sysctl -w net.ipv4.ip_forward=1
```

Команда `sysctl -a` возвращает список всех параметров ядра, а также текущие настройки этих параметров. Команда `sysctl -p` позволяет активизировать настройки `sysctl`, сохраненные в файле. В качестве имени файла обычно используется `/etc/sysctl.conf`. Синтаксис описан на справочном сайте по `sysctl.conf`. Во многих дистрибутивах (например, Debian, Fedora, Red Hat, SUSE, Ubuntu) этот файл автоматически интерпретируется и выполняется в ходе процесса Init-V.

18 Конфигурация сети

В этой главе описано, как создать на компьютере с Linux выход в Интернет или локальную сеть. В главе сначала представлен сетевой менеджер (Network Manager), помогающий подключать к Интернету ноутбуки и настольные компьютеры (LAN, WLAN, UMTS и т. д.). В следующих разделах приводится базовая информация о конфигурации сети и рассказывается о том, как выполняется статическая конфигурация сети в текстовом режиме (прежде всего, в серверных системах).

18.1. Network Manager

В настоящее время Network Manager — самый популярный инструмент для конфигурирования LAN, WLAN, ADSL, UMTS и VPN. Он применяется почти во всех дистрибутивах. За основные функции программы отвечает фоновый процесс (демон), который запускается при старте компьютера.

Графический пользовательский интерфейс сетевого менеджера выглядит поразному в зависимости от конкретной настольной системы. В этом разделе речь пойдет в основном о программах `nm-applet` и `nm-connection-editor`, применяемых в Ubuntu, в современных версиях Gnome и KDE для конфигурирования сети применяется специальный модуль системных настроек. Диалоговые окна там выглядят по-своему, но работа с системой происходит по тому же принципу, что и в Ubuntu.

Предпосылки. Для работы сетевого менеджера необходимо, чтобы эта программа получила контроль над сетевыми интерфейсами. На данном этапе конфигурация отличается от дистрибутива к дистрибутиву.

В Debian и Ubuntu необходимо убедиться, что файл `/etc/network/interfaces` содержит лишь настройки интерфейса обратной петли (loopback interface) — по умолчанию обычно так и бывает. Интерфейсы, которые должны управляться сетевым менеджером (как правило — `eth0` и `wlan0`), нельзя конфигурировать в этом файле!

```
# Файл /etc/network/interfaces (Debian, Ubuntu)
auto lo
iface lo inet loopback
```

В Fedora и Red Hat в файле `/etc/sysconfig/network-scripts/ifcfg-`*имя* должна присутствовать запись `NM_CONTROLLED=yes`. При этом *имя* — это имя устройства,

соответствующее интерфейсу (например, eth0). При установке системы на настольном компьютере или ноутбуке верной является стандартная настройка.

```
# Файл /etc/sysconfig/network-scripts/ifcfg-xxx (Fedora, Red Hat)
...
NM_CONTROLLED="yes"
```

В SUSE YaST-модуль Сетевые устройства ▸ Сетевые настройки определяет, какая именно система отвечает за конфигурацию сети — сетевой менеджер или (как обычно) YaST и ifup. По умолчанию в дистрибутиве SUSE на настольных компьютерах применяется традиционная конфигурация, а при работе с ноутбуками — сетевой менеджер.

Как деактивировать Network Manager. Если вы конфигурируете компьютер как сервер или роутер, то Network Manager следует отключить и статически выполнить конфигурацию сети.

Конфигурация

На панели Ubuntu имеется ярлык, демонстрирующий текущее состояние сети. Этот ярлык открывает меню, в котором перечислены данные об активном соединении, всех доступных точках WLAN, а также о различных командах (рис. 18.1).

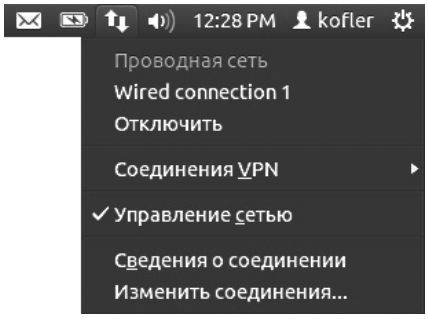


Рис. 18.1. Главное меню сетевого менеджера

LAN с DHCP (ADSL-роутер)

Простейшая ситуация, в которой применяется Network Manager, — это подключение компьютера сетевым кабелем к ADSL-роутеру, серверу LAN или другому компьютеру, на котором работает DHCP-сервер. DHCP — это протокол, по которому клиентам с локальных компьютеров присваивается конфигурация. По умолчанию Network Manager проверяет все интерфейсы LAN на предмет того, можно ли через этот интерфейс передать конфигурационную информацию по протоколу DHCP. Если это возможно, то сетевая конфигурация проходит полностью автоматически и вы будете в Сети еще до того, как войдете в систему (поскольку Network Manager активируется уже при запуске системы).

СОВЕТ

Дистрибутивы на основе Red Hat по умолчанию сконфигурированы так, что соединение не устанавливается автоматически. В таком случае выполните в меню сетевого менеджера команду Изменить соединения и найдите в конфигурационном окне соединение SYSTEM ETH0. Команда Изменить этого соединения открывает окно с подробной информацией, в котором можно установить флажок Автоматическое соединение. Сложно понять, почему здесь этот параметр не задан по умолчанию, как в других дистрибутивах.

Статическая конфигурация LAN. Статическая конфигурация LAN-соединения может потребоваться в тех случаях, когда ваш компьютер не соединен с маршрутизатором или DHCP-сервером и вам приходится самостоятельно указывать IP-адрес, маску сети, сетевой адрес и адрес сервера имен. (Все перечисленные здесь термины разъяснены в сетевом глоссарии в разделе 18.2.)

Чтобы начать конфигурацию, щелкните на значке Network Manager правой кнопкой мыши и выберите Обработка соединения. В результате вы попадете в диалоговое окно конфигурации сети.

Создание доступа к WLAN. Network Manager самостоятельно находит все расположенные поблизости сети WLAN. Когда вы впервые выбираете название WLAN в меню Network Manager, нужно указать пароль для входа в эту сеть. Обратите внимание, что в окне настройки предлагается несколько возможностей ввода ключа, например при шифровании WEP ключ можно вводить как кодовую фразу (фразу-пароль), как шестнадцатеричный код или как код ASCII. Выберите нужный вам формат. Шестнадцатеричные коды вводятся без `0x` перед ними.

При первоначальном установлении соединения в корпоративной сети, где применяется система шифрования WPA & WPA2, в сетевом менеджере необходимо выбрать УЦ-сертификат. Нужные сертификаты находятся в каталоге `/etc/ssl/certs`. То, какой именно сертификат вам требуется, зависит от конфигурации корпоративной сети. О такой конфигурации вам может рассказать системный администратор.

При последующих входах в сеть Network Manager будет устанавливать соединение самостоятельно. Для этого пароли WLAN централизованно сохраняются в системе. В некоторых системах, используемых на локальных компьютерах, при первом обращении к памяти может понадобиться ввести master-пароль. Если возникнут проблемы с изменением имеющегося пароля WLAN, запустите в Gnome программу `seahorse`. На вкладке Пароли можно изменить или удалить все пароли.

Сеть WLAN может быть сконфигурирована так, что при соединении она не сообщает своего имени. В таком случае имя не отображается в меню Network Manager. Чтобы все же установить соединение, откройте окно настроек с помощью команды меню Подключиться к скрытой беспроводной сети. Здесь самостоятельно задайте имя сети (ESSID — расширенный набор служб идентификации), а также способ шифрования.

Конфигурирование точки соединения с сетью WLAN. Если ваш компьютер имеет доступ в Интернет через интерфейс LAN и, кроме того, на нем установлен контроллер WLAN, то этот компьютер можно конфигурировать с помощью Network Manager так, чтобы он функционировал как точка доступа в сеть WLAN для других машин. (На смартфонах такой метод часто называется тетерингом от английского

слова tether — «привязывание»). Правда, этот метод действует лишь с некоторыми контроллерами WLAN, драйверы которых для Linux поддерживают работу в так называемой простой сети (режим ad-hoc).

Механизм конфигурирования прост: в меню Network Manager выбирается Создать новое беспроводное соединение, затем указывается нужный параметр. Кнопка Создать станет активной только после того, как система сочтет пароль достаточно длинным.

Но когда я пытался протестировать эту функцию на различных ноутбуках, результаты были скорее неутешительными: хотя в ходе конфигурации система не выдавала никаких сообщений об ошибках, мне не удалось использовать другие устройства в WLAN-сети компьютера с Ubuntu, как с шифрованием, так и без него.

ADSL-модем. Если вы выходите в Интернет через ADSL, то проще всего подключиться через ADSL-роутер: соедините компьютер и роутер сетевым кабелем. Network Manager самостоятельно выполнит сетевую конфигурацию.

Ситуация осложняется, если ADSL-модем подключен к компьютеру напрямую. Тонкости процесса зависят от того, какой модем вы используете и где находитесь. Network Manager поможет вам лишь в том случае, если вы работаете с модемом, который подсоединен к компьютеру сетевым кабелем (а не через USB). В зависимости от провайдера, ADSL-соединение может строиться на базе различных протоколов. Если провайдер использует протокол PPPoE, то выполните команду Обработка соединения контекстного меню, перейдите на вкладку DSL и нажмите кнопку Добавить. Теперь заполните поля Логин и Пароль. Немного везения — и у вас получится установить соединение. Если же сразу дело не пойдет, вам придется поискать на остальных не слишком понятных вкладках настройки, требуемые вашим провайдером.

Некоторые провайдеры ADSL используют вместо PPPoE протокол PPTP. В таком случае в процессе конфигурации необходимо применять вместо DSL вкладку VPN.

Сети для мобильной связи. Когда вы подключаете к компьютеру USB-модем, через несколько секунд открывается окно, в котором требуется ввести PIN-код. Если вы хотите, чтобы сетевой менеджер запомнил код, дополнительно установите флажок Автоматически разблокировать данное устройство.

После ввода PIN-кода в меню сетевого менеджера появится команда Создать мобильное широкополосное соединение. Через эту запись в меню вы попадаете в конфигурационный ассистент. В первом диалоговом окне ассистента вы выбираете вашу страну, во второй — провайдера (то есть оператора мобильной связи, предоставляющего вам выход в Интернет). Если вашего провайдера нет в списке, то нужные данные также можно ввести вручную. На последнем этапе для некоторых провайдеров можно задать режим тарификации (например, фиксированную ставку).

Любую указанную информацию позже можно изменить с помощью команд Изменить соединения ▸ Мобильные. На данном этапе также можно установить флажок Подключаться автоматически. В таком случае менеджер устанавливает соединение с широкополосной сетью без запроса, как только вы подключаете модем к компью-

теру. Но эту возможность рекомендуется применять лишь в тех случаях, когда вы пользуетесь безлимитным тарифом.

Внутренняя организация

Сетевой менеджер на уровне команд. Команда `nm-tool` выводит подробную информацию обо всех соединениях, управляемых сетевым менеджером. Кроме того, соединениями можно управлять с помощью команды `nmcli`. Таким образом, для управления сетевым менеджером можно применять сценарии, а в случае с серверными системами отпадает необходимость в графическом пользовательском интерфейсе. Следующие команды сначала выстраивают список всех соединений, известных сетевому менеджеру, после чего активизируется соединение с именем `System eth0`.

```
root# nmcli con list
NAME        UUID                                  TYP             ...
System eth0 5fb06bd0-0bb0-7ffb-45f1-d6edd65f3e03  802-3-ethernet  ...
root# nmcli con up id 'System eth0'
```

WLAN-пароли. Сетевой менеджер сохраняет пароли для всех беспроводных сетей, которые когда-либо были созданы. Пароли записываются незашифрованным текстом в файлах каталога `/etc/NetworkManager/system-connections`. Чтобы читать этот каталог, нужны права администратора, но, тем не менее, сохранение паролей в незашифрованном виде — крайне неудачная идея: https://bugzilla.redhat.com/show_bug.cgi?id=729998.

Диспетчер. При установлении или разрыве соединения можно автоматически выполнять сценарии. Эти сценарии должны создаваться в каталоге `/etc/NetworkManager/dispatcher.d/`.

Сервер имен. Современные версии сетевого менеджера (например, в Ubuntu 12.04 и выше) сконфигурированы так, что они автоматически запускают программу Dnsmasq, которая действует в качестве кэша сервера имен. Dnsmasq запоминает IP-адреса хост-имен, благодаря чему обеспечивается более быстрое разрешение адресов. Подробное описание Dnsmasq приводится в разделе 19.5.

Сетевой менеджер самостоятельно создает конфигурационный файл для Dnsmasq и сохраняет этот файл под именем `/var/run/nm-dns-dnsmasq.conf`. Если вы (из соображений безопасности) воздерживаетесь от использования локального сервера имен либо если в сети и так работает сервер имен, можно обойтись без автоматического запуска Dnsmasq. Для этого удалите в файле `/etc/NetworkManager/NetworkManager.conf` команду `dns=dnsmasq`. Базовая информация о конфигурации, специфичной для Ubuntu, приводится на сайте http://www.stgraber.org/2012/02/24/dns-in-ubuntu-12-04/.

Альтернативы сетевого менеджера

По причинам исторического характера и из-за того, что в Network Manager по-прежнему есть много недоработок, в некоторых крупных дистрибутивах, а также

в Рабочих столах Gnome и KDE предусмотрены собственные программы, упрощающие конфигурацию сети.

System-config-network (Fedora, Red Hat). Программа `system-config-network`, уже много лет входящая в состав дистрибутивов Fedora и Red Hat, помогает обеспечить доступ к сетям через соединения LAN, WLAN, ADSL, ISDN или через аналоговый модем (рис. 18.2). При работе с адаптерами WLAN `system-config-network` поддерживает лишь ненадежное WEP-шифрование, поэтому рекомендую использовать вместо нее Network Manager.

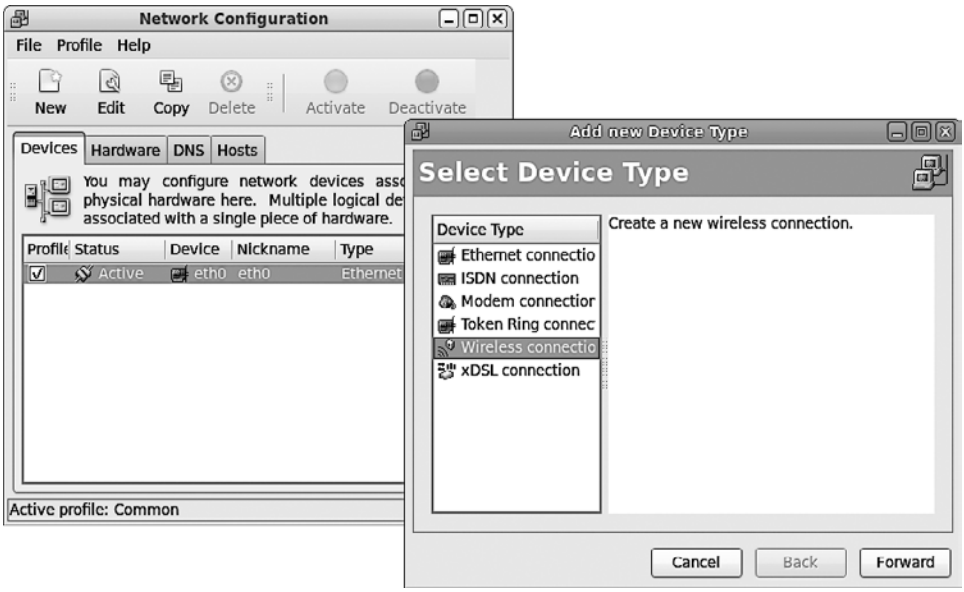


Рис. 18.2. Конфигурация сети с помощью system-config-network

Если с отдельными сетевыми интерфейсами вы *не* хотите использовать сетевой менеджер, то для данного интерфейса нужно снять флажок Управляется сетевым менеджером. Кроме того, необходимо гарантировать, что во время запуска системы выполняется Init-сценарий `/etc/init.d/network`, чего по умолчанию не происходит:

```
root# service network start          (Запускаем сейчас)
root# chkconfig --level 35 network on (Автоматически запускаем в будущем)
```

YaST (SUSE). При работе на ноутбуках SUSE автоматически активизирует сетевой менеджер, но при использовании настольными компьютерами продолжает применять классический вариант управления сетевыми функциями — с помощью YaST и `ifup`. Конфигурация осуществляется в YaST-модуле Сетевые устройства ▸ Настройки сети. На вкладке Глобальные параметры можно выбрать одну из команд: Традиционный метод с применением ifup или Пользовательское управление с помощью сетевого менеджера.

Если вы решите воспользоваться традиционным методом, то YaST выведет на вкладке Обзор все обнаруженные сетевые интерфейсы. Теперь конкретные интер-

фейсы можно изменить, нажав кнопку Изменить. По умолчанию YaST конфигурирует все Ethernet-интерфейсы так, что они получают IP-адрес и все другие конфигурационные настройки по DHCP от (ADSL-)маршрутизатора локальной сети. В случае с WLAN-интерфейсами потребуется еще второй этап, на котором задаются сетевые имена (ESSID), методы аутентификации (WEP или WPA) и пароль.

Gnome-ppp (Gnome). Аналоговые модемы не подходят для работы с современными сайтами и поэтому обречены на вымирание. И все же такие модемы — те самые незаменимые устройства, которые позволяют прочитать электронное сообщение в старой гостинице. Если Linux распознает ваш аналоговый модем, то установить соединение проще всего будет, используя программу `gnome-ppp`, разработанную для создания соединений через аналоговые модемы в Gnome и KDE, или с помощью другой программы — KPPP. Поскольку аналоговые модемы выходят из употребления, эти программы по умолчанию не устанавливаются. Кроме того, многие встроенные аналоговые модемы — это так называемые винмодемы, то есть устройства, для которых есть только драйверы Windows. Следовательно, такие модемы нельзя использовать с Linux, а если и можно, то с большим трудом.

Работать с `gnome-ppp` достаточно просто: вы указываете три ключевых параметра для доступа к Интернету — логин, пароль и телефонный номер — и нажимаете кнопку Подключение. В идеальном случае соединение устанавливается с ходу. Если сразу это не получится, откройте нажатием кнопки Конфигурация диалоговое окно Установка и настройте в нем остальные параметры. При выборе типа модема для большинства встроенных модемов применяется вариант Аналоговый модем (то же касается модемов, подключенных через последовательные интерфейсы) или USB-Модем, если модем подключен к компьютеру USB-кабелем.

KPPP (KDE). При работе с KPPP начните настройку вашего модема и соединения с Интернетом (логин, пароль, телефонный номер), нажав кнопку Создать. Если вы полагаете, что вам, возможно, придется менять провайдеров, отдельно сконфигурируйте данные доступа и модем. KPPP поддерживает значительно больше параметров, чем `gnome-ppp`, из-за этого конфигурационные диалоговые окна программы KPPP весьма запутанны.

Pppoeconfig. В некоторых дистрибутивах (например, Debian и Ubuntu) для настройки конфигурации модемов ADSL предоставляется команда `pppoeconf`. Она запускается от имени администратора в окне терминала. Команда просматривает все сетевые интерфейсы и ищет ADSL-модем. Предварительная конфигурация сетевого интерфейса не требуется.

Когда модем найден, введите имя пользователя и пароль. Последующие запросы программы, касающиеся автоматической конфигурации DNS, а также конфигурации MSS (системы коммутации региональной сети) можете просто подтверждать.

Например, при работе с немецким провайдером T-online имя пользователя состоит из трех элементов: двенадцатизначного номера абонента `A`, номера T-online `T` и так называемого сопользовательского номера (обычно `0001`). В результате получаем следующую последовательность символов:

```
AAAAAAAAAAAATTTTTTTTTT#0001@t-online.de
```

Если T состоит из 12 символов или более, # не используется.

Наконец, программа спрашивает вас, следует автоматически устанавливать ADSL-соединение при запуске системы или соединение будет задаваться вручную. Первый вариант целесообразно использовать, если у вас безлимитный тариф на Интернет и вы все время хотите оставаться в Сети. Если вы выберете второй вариант, доступ к ADSL нужно будет активизировать, а затем снова отключать с помощью следующих команд:

```
root# pon dsl-provider
...
root# poff dsl-provider
```

18.2. Основы работы с сетью и глоссарий

В этом разделе рассмотрены основы конфигурации сети. Информация касается не только подключения к LAN (локальной сети), но и конфигурации WLAN, ADSL и модема.

Глоссарий

TCP/IP. Во всех распространенных операционных системах поддерживается *сетевой протокол TCP/IP*. Он регулирует обмен информацией между компьютерами, как в локальных сетях (LAN, интранет), так и в Интернете. Поэтому для многих сетевых служб неважно, где именно находится другой компьютер из данной сети — на расстоянии пяти метров или в Японии. Во втором случае только немного снизится скорость.

Интернет-протокол (IP) служит основой для *протокола управления передачей* (TCP). Общий протокол TCP/IP выполняет две важные задачи: он идентифицирует любой компьютер по уникальному номеру (IP-адресу) и обеспечивает надежную передачу данных, то есть отвечает за то, чтобы данные, отправленные по сети, пришли именно к адресату, а не куда-то еще. Данные передаются в виде небольших пакетов.

Сетевые функции TCP/IP требуются вам даже тогда, когда компьютер не работает в сети и у вас нет ни сетевого подключения, ни модема! Многие программы Linux используют этот протокол как раз для внутрисистемной передачи данных. Поэтому вам в любом случае потребуется установить петлевой (кольцевой) интерфейс. Это необходимо делать во всех дистрибутивах.

UDP и ICMP. Наряду с TCP существуют еще два протокола, играющих в Интернете очень важные роли: UDP и ICMP. UDP — это *протокол пользовательских датаграмм*. Он обеспечивает так называемую ненадежную передачу пакетов. «Ненадежный» в данном контексте означает, что отправитель и получатель не должны постоянно поддерживать сетевое соединение и обмениваться пакетами с информацией. Поэтому при работе с UDP может случиться так, что какие-то пакеты не дойдут до адресата или дойдут не в той последовательности, в которой были отправлены. Восстановлением целостности данных в этом случае занимается полу-

чатель, а не система. У UDP по сравнению с TCP есть определенное преимущество — этот протокол (по крайней мере в некоторых сферах) более эффективен и быстрее реагирует при передаче данных, так как его КПД выше. Например, этот протокол используется со службами DNS и NFS.

ICMP — *протокол управления сообщениями в сети*. В принципе этот протокол был разработан не для обмена данными между программами, а для передачи управляющих кодов и кодов ошибок для TCP/IP. В частности, протокол ICMP применяется вспомогательным инструментом `ping`.

Порты. Каждый IP-пакет относится к определенной категории по номеру порта. Поэтому на стороне получателя становится проще упорядочивать пришедшие пакеты. Большинство интернет-приложений (WWW, FTP, электронная почта и т. д.) работают со специально выделенными для них портами.

PPP. Если соединение с Интернетом осуществляется без использования инфраструктуры локальной сети, а непосредственно через модем или ISDN-карту, то обычно применяется *протокол двухточечного соединения (PPP)*. Этот протокол позволяет передавать данные TCP/IP по телефонной линии, по ISDN или ADSL. PPP также можно использовать для создания *виртуальных частных сетей*.

Хост-имя и доменное имя. Пусть IP-адреса и очень практичны при работе с компьютером, но запоминаются они очень плохо. Поэтому компьютер параллельно можно идентифицировать и по комбинации хост-имени и доменного имени.

Хост-имя — это имя самого компьютера. *Доменное имя* обозначает частную сеть, в рамках которой может быть запрошен компьютер, и может состоять из нескольких частей.

В локальной сети хост-имена и доменные имена служат в первую очередь для того, чтобы пользователям было легче их запоминать. Например, мои тестовые компьютеры названы так же, как и планеты Солнечной системы (jupiter и др.), а в качестве доменного имени используется sol. Таким образом, полное имя — jupiter.sol.

СОВЕТ

В качестве хост-имен не следует использовать названия производителей компьютеров, их владельцев или проектов, выполняемых в настоящее время, так как в таком случае легко запутаться. Указывайте короткие и хорошо запоминающиеся названия зверей, растений, планет, рек или какие-то другие понятия, которые просто запомнить. Не применяйте в названия специальных символов из национальных алфавитов.

Никогда не задействуйте в качестве хост-имени localhost! Оно особенное и считается названием отдельной сети (полностью уточненное имя). Этому названию всегда присвоен адрес 127.0.0.1, соответствующий петлевому интерфейсу, независимо от остальных параметров конфигурации сети.

При выборе доменного имени ограничений еще больше. Это имя должно соответствовать тому доменному имени, которое уже используется в сети. Только создавая новую локальную сеть, вы можете свободно выбирать доменное имя.

Если компьютер с Linux действует как открытый сервер, видимый в Интернете и служащий для работы с Сетью, электронной почтой или другими службами, то вы должны зарегистрировать желаемое имя у провайдера интернет-услуг или в Сетевом Информационном Центре (кратко — NIC). Например, домены

зоны .de регистрируются в http://www.denic.de, а домены .com, .net и .org — в http://www.corenic.org.

IP-адреса или IP-номера. IP-адреса предназначены для того, чтобы однозначно идентифицировать компьютер в сети. Типичный IP-адрес компьютера в локальной сети — 192.168.0.75. Базовая информация по IP-адресам сообщается в следующем подразделе.

MAC-адрес. MAC-адрес (MAC — Media Access Control, управление доступом к среде) — это уникальный ID-номер, который есть у каждого Ethernet-контроллера. Номер MAC обеспечивает идентификацию сетевого контроллера еще до присвоения ему IP-адреса. В частности, адреса MAC используются при работе с протоколом DHCP (см. раздел 19.5).

Интерфейс. IP-адрес обозначает не компьютер, а IP-интерфейс. Часто у компьютера есть несколько интерфейсов с различными IP-адресами. Обычно применяются петлевой интерфейс (127.0.0.1), один или несколько Ethernet-интерфейсов, а также, возможно, PPP-интерфейс, обеспечивающий доступ к Интернету через модем, ISDN или ADSL.

Если речь идет только об *одном* IP-адресе, то обычно подразумевается адрес, по которому компьютер запрашивается в локальной сети или Интернете. Как правило, это IP-адрес интерфейса Ethernet, присваиваемый хост-имени и доменному имени и являющийся уникальным в пределах сети.

Название интерфейса. Внутри системы Linux всем интерфейсам присваиваются названия (имена). Обычно петлевой интерфейс имеет название `lo`, интерфейсы Ethernet — названия `eth0`, `eth1` и т. д. и `pppn` — для интерфейсов PPP.

В Fedora при именовании Ethernet-интерфейсов используется иная номенклатура. Сетевые интерфейсы, интегрированные в материнскую плату, получают имена вида `em`*n*, причем, нумерация начинается с 1. Сетевые устройства, подключаемые через PCI-шину, получают имена вида `ps`*pn*, где `s` означает PCI-ячейку, а *n* — номер порта (нумерация также начинается с 1).

Петлевой интерфейс. Петлевой интерфейс играет особую роль: он обеспечивает применение сетевого протокола с локальными службами, то есть обмен информацией внутри компьютера. Это, возможно, звучит абсурдно, но такой механизм применяется многими простейшими командами Linux. Причина такова: обмен информацией между большинством командами протекает на базе сетевого протокола, и при этом неважно, остаются данные в пределах локального компьютера или их обработка продолжается на удаленном компьютере. Так работает, например, система печати (CUPS), выполняющая свои задачи, как на локальном компьютере, так и на других компьютерах в сети.

За петлевым интерфейсом закреплен IP-адрес 127.0.0.1. Во всех дистрибутивах настройка петлевого интерфейса осуществляется автоматически, даже если никакой сетевой конфигурации не происходит.

Маска сети, сетевой адрес и широковещательный адрес. Протяженность локальной сети выражается двумя или тремя масками. *Маски* — это четырехчастные группы цифр, используемые внутри системы как конфигурации битов для IP-адресов. Если локальная сеть включает в себя все номера 192.168.0.*n*, то соответствующая маска сети будет 255.255.255.0, сетевой адрес — 192.168.0.0, а широковещательный адрес — 192.168.0.255 (во многих конфигурационных программах не

требуется вводить широковещательный адрес, так как он автоматически выводится из двух других адресов).

Полученная сеть будет обозначаться как 192.168.0.0/255.255.255.0 или коротко — 192.168.0.0/24 (в кратком варианте записи указывается количество двоичных единств в маске сети). Это означает, что два компьютера, имеющие IP-адреса 192.168.0.71 и 192.168.0.72, могут напрямую обмениваться информацией в этой сети (так как IP-номера в адресном пространстве маски сети соответствуют друг другу). Максимальное количество компьютеров, которые могут одновременно обмениваться данными в рамках этой сети, составляет 254 (от .1 до .254) — номера .0 и .255 зарезервированы.

Шлюз. Это компьютер, стоящий на стыке двух сетей (зачастую между локальной сетью и Интернетом). Чтобы ваш компьютер с Linux, работающий в локальной сети, имел выход в Интернет, при конфигурации необходимо указать *адрес шлюза*.

Итак, адрес шлюза обозначает один из компьютеров в локальной сети, например 192.168.0.1. Этот компьютер играет особую роль, так как через него (например, с помощью ADSL) осуществляется соединение с Интернетом. Таким образом, весь интернет-трафик локальной сети проходит через компьютер-шлюз.

Сервер имен. Это программа, преобразующая имена компьютеров или интернет-адреса (например, www.yahoo.com) в IP-адреса. В небольших сетях имена и номера часто соотносятся по таблице (файл `/etc/hosts`). В Интернете эту задачу выполняют специальные базы данных. Вместо термина *сервер имен* часто применяется аббревиатура *DNS (Domain Name Server, сервер доменных имен)* или термин *службы*.

Если вы хотите просмотреть в браузере сайт yahoo.com, то сначала система свяжется с сервером имен, чтобы узнать номер сервера www.yahoo.com. Когда эта задача будет выполнена, устанавливается связь с указанным IP-адресом.

DHCP. *Протокол динамической настройки хостов* (Dynamic Host Configuration Protocol, DHCP) часто используется в локальных сетях для централизованного администрирования сети. Чтобы не приходилось отдельно настраивать для каждого компьютера IP-адрес, шлюз, сервер имен и т. д., один из компьютеров конфигурируется как DHCP-сервер (см. раздел 19.5). Все остальные компьютеры в сети при запуске системы выходят на связь с DHCP-сервером и спрашивают его, какие настройки необходимо применить. Таким образом, работа по конфигурированию клиента сводится к минимуму.

IP-адреса

Как уже было сказано выше, IP-адреса предназначаются для идентификации компьютеров в сети. Это правило касается как локальных сетей, так и Интернета. В данном подразделе будет рассмотрена базовая информация, касающаяся применения IP-адресов.

Теоретически существует 2564, то есть около 4 миллиардов IP-адресов. На самом деле таких адресов доступно гораздо меньше, так как часть из них зарезервирована (в том числе все IP-адреса, начинающиеся с .0 или .255). Кроме того, раньше IP-адреса раздавались очень большими пакетами.

В эпоху бурного роста Интернета становится все сложнее выполнять требование соблюдения во всем мире уникальности IP-адресов для всех имеющихся компьютеров. До тех пор пока не закрепился формат IPv6 (это новая версия интернет-протокола, которая наряду со многими другими улучшениями позволяет значительно расширить адресное пространство), IP-адреса останутся весьма ограниченным ресурсом.

IP-адреса в Интернете

Если вы хотите подключить свой веб-сервер к Интернету, вам потребуется доменное имя, действующее в Глобальной сети (например, moyafirma.ru), а также собственный IP-адрес.

И домен, и адрес проще всего получить у провайдера или в сетевом информационном центре (Network Information Center, NIC).

Для частных пользователей и небольших организаций это, как правило, совсем не обязательно. В локальной сети используются IP-адреса так называемого *частного адресного пространства*. Связь с Интернетом обеспечивается провайдером, который предоставляет (на время соединения) IP-адрес, действительный во всем мире.

Если вам к тому же требуется иметь в Интернете постоянное представительство, то есть собственный сайт, его также можно получить с помощью провайдера услуг. Компьютер с сервером и вашими HTML-документами находится у провайдера (а не у вас дома), за IP-адрес опять же отвечает провайдер. Такой метод имеет определенные плюсы — вам не требуется постоянное соединение с Интернетом, а сайт доступен в Интернете все время.

Резюмирую: собственный, уникальный и действительный во всем мире IP-адрес вам нужен лишь в том случае, если ваш компьютер должен постоянно быть доступен в Интернете и для этого у вас установлено непрерывное соединение с Сетью (например, через выделенную линию). Как правило, такое соединение требуется лишь крупным фирмам или, например, университетам.

IP-адреса в локальных сетях

Компьютеры, расположенные в локальных сетях, в Интернете, как правило, невидимы. Это означает, что при правильной конфигурации такие компьютеры могут работать в Интернете, но в то же время остаются защищены от неконтролируемого доступа из Интернета. Таким образом, IP-адреса, действующие в локальной сети, должны быть уникальны только в пределах этой сети, но не в глобальном масштабе.

Поскольку IP-адресов все больше не хватает, такая экономия представляется очень привлекательной. На данный момент в адресном пространстве номеров IP для локальных сетей зарезервировано три диапазона:

- 10.0.0.0–10.255.255.255;
- 172.16.0.0–172.31.255.255;
- 192.168.0.0–192.168.255.255.

Первый диапазон позволяет построить очень большую локальную сеть (теоретически — с 16 миллионами компьютеров, этого достаточно даже для очень круп-

ных фирм). Во втором диапазоне мы имеем дело, собственно, с 16 подсетями, каждая из которых включает около 65 000 адресов (например, от 172.23.0.0 до 172.23.255.255). Третий диапазон состоит из 256 малых подсетей. Одна из таких сетей — от 192.168.75.0 до 192.168.75.255.

Неважно, в какой из подсетей строится ваша локальная сеть; такой метод исключает возможность возникновения адресных конфликтов с «настоящими» IP-адресами.

Как правило, пользоваться функциями Интернета требуется и в локальной сети (например, просматривать сайты). Чтобы это было возможно, один из компьютеров локальной сети должен быть сконфигурирован как интернет-шлюз. Этот компьютер обеспечивает соединение с Интернетом (через ADSL, ISDN, модем или иным способом) и переадресовывает все запросы, поступающие из локальной сети. Кроме того, шлюз должен заменять IP-адреса локальной сети IP-адресами, действительными в глобальном масштабе. Эта функция называется *маскарадинг*. Она рассмотрена в главе 19.

Динамические IP-адреса

Итак, понятно, что для идентификации компьютера или сети требуется IP-адрес. Но откуда компьютеру будет известно, какой IP-адрес он должен применять? Простейший способ сообщить об этом — прямо задать IP-адрес при конфигурации. В небольших локальных сетях обычно так и делается. Например, первый компьютер в сети получает адрес 192.168.0.1, следующий — 192.168.0.2 и т. д. Адрес сохраняется в файле `/etc/hosts`.

Размеры сетей увеличиваются, и проводить такую децентрализованную конфигурацию становится все сложнее. Чтобы такой конфигурации не требовалось, часто применяются динамические IP-адреса. Для этого один из компьютеров в сети необходимо сконфигурировать как DHCP-сервер (выше было указано, что DHCP — это протокол динамической настройки хостов). Все остальные компьютеры перед началом использования сетевых функций устанавливают соединение с DHCP-сервером, и он присваивает им IP-адреса. Часто задачи DHCP-сервера выполняет ADSL-роутер.

Подобный метод имеет два основных преимущества: во-первых, всей сетью можно управлять централизованно (а не настраивать параметры на каждом компьютере отдельно). Во-вторых, работа по администрированию клиентского компьютера сводится к нулю. Чтобы подключить клиентский компьютер к сети, нужно просто указать имя компьютера и установить флажок DHCP. Все остальные данные (сам IP-адрес компьютера, IP-адреса DNS и шлюза и т. д.) передаются через DHCP.

Кроме того, есть еще и третье преимущество, которое, правда, в большей степени касается провайдеров интернет-услуг: поскольку IP-адреса присваиваются динамически, а в сети обычно бывает активна только часть компьютеров, достаточно сравнительно небольшого количества IP-адресов, чтобы обеспечить ими всех нуждающихся абонентов. Всякий раз, когда клиент провайдера выходит в сеть через модем или ISDN, ему предоставляется ближайший свободный IP-адрес.

Один компьютер с несколькими IP-адресами

Как правило, один компьютер имеет несколько IP-адресов. Ранее, когда мы говорили об *одном* IP-адресе, имелось в виду, что один IP-адрес присваивается интерфейсу одного сетевого контроллера. Это номер, по которому компьютер идентифицируется в сети (что, по сути, означает: IP-адрес имеет не компьютер, а интерфейс сетевого контроллера, установленного на этом компьютере).

Кроме того, любой компьютер с UNIX/Linux доступен по адресу 127.0.0.1 или под именем localhost. Это адрес уже упоминавшегося выше петлевого интерфейса, который предназначен для работы только с трафиком локальной сети. Для того чтобы проверить, работает ли этот механизм, нужно просто выполнить команду `ping`:

```
user$ ping localhost
PING localhost (127.0.0.1) 56(84) bytes of data.
64 bytes from localhost (127.0.0.1): icmp_seq=1 ttl=64 time=0.049 ms
64 bytes from localhost (127.0.0.1): icmp_seq=2 ttl=64 time=0.040 ms
64 bytes from localhost (127.0.0.1): icmp_seq=3 ttl=64 time=0.041 ms

--- localhost ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 1999ms
rtt min/avg/max/mdev = 0.040/0.043/0.049/0.006 ms
```

Команда `ping` отсылает по указанному адресу небольшие пакеты с данными и измеряет, сколько времени пройдет до подтверждения прибытия пакетов. Команда `ping localhost` должна работать и в том случае, если на компьютере отсутствует сетевой контроллер!

Наконец, возможна ситуация, когда на компьютере установлено несколько сетевых контроллеров. Каждый контроллер считается отдельным интерфейсом и поэтому имеет собственный IP-адрес. Кроме того, отдельный интерфейс задействуется при PPP-соединении, создаваемом с помощью ISDN-карты или при выходе в Интернет через модем. Такому интерфейсу также присваивается IP-адрес, причем этот адрес, как правило, назначается провайдером интернет-услуг (в данном случае речь идет о динамическом IP-адресе).

Несколько сетевых контроллеров может быть установлено на компьютере и в том случае, если он должен связать две подсети с различными адресными пространствами. Такой компьютер называется *роутером*. Примером роутера является интернет-шлюз локальной сети. При создании соединения с Интернетом компьютер-шлюз имеет как минимум три IP-адреса: адрес петлевого интерфейса 127.0.0.1, адрес, действующий в локальной сети и, наконец, выделенный провайдером интернет-услуг глобальный адрес во Всемирной паутине. (Строго говоря, сам шлюз, как правило, не занимается роутингом, а только выполняет функцию маскарадинга. Это тонкое отличие более подробно описано в главе 19.)

Статическая конфигурация IP-адреса

Если в локальной сети нет DHCP-сервера, то при конфигурации сети IP-адрес сетевого интерфейса потребуется настроить статически. Какой IP-адрес нужно применять в таком случае?

- **Ваш компьютер не входит в состав локальной сети.** Не считая петлевого интерфейса 127.0.0.1, вам на данный момент не требуется никакого IP-адреса (это правило действует и в том случае, если позже компьютер подключается к Интернету через модем/ISDN/ADSL). Работа по конфигурации сводится к указанию доменного имени и хост-имени.
- **Ваш компьютер входит в состав локальной сети.** В таком случае IP-адрес должен принадлежать к адресному пространству, выделенному для данной сети (например, 192.168.0.*) и быть уникальным в пределах этой сети.
- **Ваш компьютер должен служить основой для локальной сети.** Используйте частное адресное пространство (например, 192.168.0.*) и присвойте компьютеру предназначенный для этого IP-адрес.

IPv6

До сих пор мы говорили о четвертой версии IP-адресов (IPv4). На сегодняшний момент Интернет работает на основе этой версии. Однако уже в течение нескольких лет ощущается нехватка IP-адресов, причем со временем она становится все более серьезной. Кроме того, данный протокол имеет некоторые функциональные недостатки и плохо приспособлен для использования различных новых технологий, играющих в Интернете все более значимую роль (в частности, IP-телефонии, потокового аудио и видео).

Адреса IPv6. В новой версии IP-адресов (IPv6) эти недостатки должны быть устранены. Самое примечательное и наиболее очевидное для администраторов изменение заключается в том, что теперь IP-адреса будут иметь длину 128 бит (а не 32, как в IPv4). При записи в традиционном формате IP-адрес будет выглядеть так:

```
121.57.242.17.122.58.243.18.19.123.59.20.244.124.60.245
```

Очевидно, что такие записи неудобны на практике. Ради экономии места адреса IPv6 будут разбиваться символом : на группы шестнадцатеричных чисел (не более восьми групп), например, так:

```
abcd:17:2ff:12aa:2222:783:dd:1234
```

Чтобы не нужно было писать лишнего, используется символ ::, который является сокращенной формой для нескольких нулевых групп:

```
abcd:17:0:0:0:0:dd:1234 → abcd:17::dd:1234
0:0:0:0:0:783:dd:1234 3 → ::783:dd:1234
```

Для localhost существует еще более компактная запись — ::1.

При отображении адресов IPv4 в формате IPv6 первые шесть групп являются нулевыми. Оставшиеся две группы можно записывать не только в шестнадцатеричной, но и в более привычной десятичной системе:

```
Адрес IPv4: ::110.111.112.113
```

В рамках перехода на IPv6 в течение нескольких лет будут совместно использоваться IPv4 и IPv6. Существуют различные методы, позволяющие передавать пакеты IPv6 по сетям IPv4 и наоборот.

IPv6 и Linux. В принципе, ядро Linux приспособлено к работе с IPv6, уже начиная с версии 2.2, но только в версии 2.6 и выше поддержка IPv6 считается окончательно доработанной. Большинство сетевых приложений уже совместимо с IPv6.

Поскольку формат IPv6 еще не закрепился в Европе (даже частично), вся информация, сообщаемая в этой книге, касается только IPv4. Если вы хотите более подробно изучить IPv6 и особенности работы с этим протоколом в Linux, рекомендую начать со следующих сайтов:

- http://www.worldipv6launch.org/;
- http://www.ipv6.org/;
- http://ru.wikipedia.org/wiki/IPv6;
- http://www.faqs.org/rfcs/rfc1752.html.

Глоссарий по стандартам WLAN

При описании беспроводных сетей применяется множество сокращений. Чаще всего используется аббревиатура WLAN (Wireless Local Area Network, беспроводная локальная сеть), несколько реже — русский вариант БЛВС (беспроводная локальная вычислительная сеть). Кроме того, часто задействуется термин Wi-Fi (от англ. Wireless Fidelity — «беспроводная точность воспроизведения»). Правда, иногда речь идет об «альянсе Wi-Fi» — консорциуме производителей, который занимается вопросами совместимости, связанными с WLAN.

В этом разделе кратко обобщена терминология, касающаяся WLAN. Если вы уже знакомы с основами WLAN, можете перейти прямо к подразделу «Поддержка WLAN в Linux» этого раздела, где рассматриваются функции WLAN, специфичные для Linux.

Стандарты. Существует большое количество стандартов WLAN. Все они определены IEEE (институтом инженеров по электротехнике и радиоэлектронике) и начинаются с номера 802.11. Буквы, следующие за этим номером, в хронологическом порядке указывают на новые версии или варианты стандарта. Далее перечислены некоторые стандарты WLAN.

- **802.11.** Первый стандарт 802.11 описывал радиочастоту 2,4 ГГц. Максимальная брутто-частота передачи данных составляла около 2 Мбит/с. Этот стандарт сегодня уже не играет никакой роли.
- **802.11a.** При применении данного стандарта радиочастота составляет 5,2 ГГц, а брутто-частота передачи данных может достигать 54 Мбит/с. Стандарт 802.11a закрепился только в США.
- **802.11b.** Оборудование стандарта 802.11b вещает на частоте 2,4 ГГц. Брутто-частота передачи данных ограничена 11 Мбит/с и не отвечает современным требованиям.
- **802.11g.** Этот стандарт развился из 802.11b и совместим с ним. Радиочастота, как и в 802.11b, составляет 2,4 ГГц, а брутто-частота передачи данных возросла до 54 Мбит/с.

- **802.11i.** Данное дополнение к 802.11a/b/g определяет механизм шифрования и аутентификации WPA2.
- **802.11n.** Развился на основе 802.11a, b и g и не только имеет гораздо более высокую брутто-частоту передачи (до 600 Мбит/с), но и отличается большей дальностью действия. Оба указанных улучшения были достигнуты благодаря одновременному применению нескольких антенн, приемников и передатчиков (технология «многоканальный вход — многоканальный выход», или MIMO). 802.11n совместим с вариантами 802.11a, b и g. Если в сети есть хотя бы один элемент, работающий с иным стандартом, чем 802.11n, этот элемент влияет на скорость во всей сети.

Брутто и нетто. Указываемая в проспектах брутто-частота передачи данных (например, 54 Мбит/с для 802.11g) часто выглядит многообещающе. Однако за вычетом издержек, связанных с конкретным протоколом, действительная частота уменьшается более чем наполовину. Этого не избежать даже в тех случаях, когда по беспроводной сети связываются всего два абонента, а физически радиосвязь устанавливается во вполне приемлемых условиях (на небольшом расстоянии, без препятствий и т. д.), к тому же в сети нет абонентов, которые работали бы с оборудованием, использующим устаревший стандарт WLAN.

Оборудование WLAN. Все современные ноутбуки уже оснащены контроллером WLAN. Радиомосты WLAN, а также специальные точки доступа и роутеры — это отдельные внешние устройства, подключаемые к локальной сети или ADSL-модему с помощью Ethernet-кабеля.

- Радиомост WLAN соединяет отдельное устройство LAN с WLAN. Таким образом, мост выполняет ту же функцию, что и WLAN-карта, только подключение к компьютеру осуществляется через Ethernet, а не через PCI, PCMCIA или USB.
- Точка доступа (Access Point) — это простейший механизм, позволяющий предоставить доступ во WLAN для нескольких WLAN-клиентов. Точка доступа, как и мост, подсоединяется Ethernet-кабелем к сетевому серверу или к сетевому концентратору. Она несколько раз в секунду посылает сигнал (маячок), предназначенный для того, чтобы другие WLAN-устройства, находящиеся в радиусе действия, могли распознать точку доступа. В отличие от радиомоста, точка доступа обычно поддерживает дополнительные режимы WLAN и может одновременно обмениваться информацией с несколькими клиентами (то есть различия между радиомостом и точкой доступа касаются в основном программной составляющей и в меньшей степени — аппаратной).
- WLAN-роутер подключает целую сеть (LAN и WLAN) к Интернету. Обычно исходным пунктом соединения является DSL-модем с Ethernet-выходом или сервер локальной сети (существуют и WLAN-роутеры с интегрированным ADSL-модемом, которые называются *шлюзами*).

  Обычно роутер состоит из точки доступа и небольшого концентратора, рассчитанного на 4–8 Ethernet-устройств. Внутреннее программное обеспечение роутера управляет доступом к Интернету и работой локальной сети. Как правило, роутер выполняет функции трансляции сетевых адресов (NAT), сервера DHCP, имеет простой брандмауэр и т. д. (эти функции подробно описаны в главе 19).

Как правило, конфигурация устройств WLAN осуществляется в браузере. Для этого устройства предоставляют специальные веб-страницы, облегчающие конфигурацию и расположенные по определенному IP-адресу (например, http://192.168.0.1). Однако обратите внимание и на то, что отдельные устройства WLAN можно сконфигурировать только с помощью специальной установочной программы, работающей лишь в Windows. Это касается, в частности, мостов WLAN. Разумеется, в Linux такие устройства можно использовать лишь в ограниченной мере.

Параметры WLAN-соединения

Если вы устанавливаете соединение между двумя устройствами WLAN, то потребуется настроить различные параметры. Кратко рассмотрим эти параметры.

Сетевой режим. Компоненты сети WLAN могут различными способами обмениваться данными друг с другом. Опишем основные режимы.

- *Инфраструктурный режим* (иногда называемый также управляемым, или ведомым (managed mode)) обеспечивает обмен информацией с центральной точкой доступа. Иными словами, сеть имеет звездчатую форму. Обычно такой центральной точкой является или точка доступа, или WLAN-роутер, но это может быть и сконфигурированный соответствующим образом компьютер.
- Устройство WLAN точки доступа работает в *ведущем режиме* (master mode) (иначе говоря, в инфраструктурном режиме функционируют компьютеры-клиенты, а в ведущем режиме работает сервер сети WLAN).
- В *режиме простой сети* (ad-hoc mode) любое устройство WLAN обменивается информацией с любым другим устройством WLAN, находящимся в зоне действия передатчика.

SSID или ESSID. Сокращения SSID (Service Set Identification, набор служб идентификации сети) или ESSID (Extended SSID, расширенный набор служб идентификации сети) обозначают обычные последовательности символов, которые служат названием беспроводной сети. Устройства WLAN могут обмениваться информацией лишь при условии, что их SSID совпадают. Таким образом, с помощью различных SSID можно разделить две различные сети WLAN, функционирующие вблизи друг друга.

В качестве последовательности символов SSID часто задается название производителя, поэтому обмен информацией между устройствами одного производителя часто настраивается с ходу, а для устройств различных производителей сначала нужно установить общую последовательность символов SSID.

Некоторые карты WLAN автоматически осуществляют конфигурацию SSID. Обратите внимание, что при указании последовательности символов SSID важен регистр!

NWID. В сети WLAN с одинаковым SSID может быть много подсетей, которые различаются по показателю NWID. На практике такая ситуация встречается редко, поэтому некоторые конфигурационные программы вообще не работают с NWID.

Иногда вместо NWID употребляется термин *домен*, но он только вносит путаницу. NWID не имеет ничего общего с привычными доменными именами IP-адресов.

Канал. В рамках каждой полосы частот, предусмотренной тем или иным стандартом 802.11*x*, есть много поддиапазонов (каналов), по которым информация может рассылаться параллельно. В инфраструктурном режиме адаптеры WLAN сами распознают канал, применяемый точкой доступа. Специально настраивать каналы необходимо лишь в том случае, когда имеет место интерференция нескольких сетей WLAN.

Ключ WEP/WPA. В целях безопасности связь по сети WLAN должна быть защищена от перехвата данных. В зависимости от того, по какому стандарту работает оборудование WLAN, могут применяться методы WEP, WPA, а лучше всего WPA2 (эти методы подробно рассмотрены в следующем подразделе). В ходе конфигурации контроллера WLAN вы указываете ключ. Обратите внимание — в некоторых конфигурационных программах перед названием ключа необходимо ставить `0x`, чтобы ввести ключ в шестнадцатеричной системе!

Безопасность WLAN

В принципе сеть WLAN можно использовать и без шифрования данных. Но в этом случае любой, кто будет находиться в зоне действия передатчика, сможет пользоваться сетью и считывать всю передаваемую в ней информацию. Таким образом, обмен данными без шифрования — не что иное, как грубая халатность!

WEP

Для шифрования передаваемых данных в первых поколениях WLAN использовался метод WEP (Wired Equivalent Privacy, конфиденциальность на уровне проводных сетей). При этом данные зашифровываются с помощью 40- или 104-битного ключа (часто говорят о 64- или 128-битном шифровании, но оставшиеся 24 разряда не используются непосредственно в целях шифрования).

Как правило, ключ WEP указывается в виде шестнадцатеричного числа (от 10 до 26 разрядов, в зависимости от количества бит в ключе). Поскольку при вводе 26-значного числа вручную легко допустить ошибку, во многих конфигурационных инструментах предусмотрена возможность создания ключа из фразы-пароля (пароля, состоящего из нескольких слов). Процесс генерирования ключа зависит от производителя. Таким образом, один и тот же пароль может реализовываться на оборудовании различных производителей в виде разных ключей. При возникновении сомнений вам не остается ничего другого, кроме как указать ключ вручную.

В ходе конфигурации WEP можно задать до четырех ключей. В действительности всегда будет использоваться только один. Однако в управлении четырьмя ключами есть определенный плюс — при смене сети WLAN вам не придется заново вводить целый ключ, а нужно будет всего лишь активизировать другой из четырех ключей.

ВНИМАНИЕ

Алгоритм WEP оказался ненадежным, так как в нем имеется несколько принципиальных недоработок! Даже 104-разрядный ключ можно узнать за несколько минут, просто перехватывая весь трафик WLAN. Программы, предназначенные для взлома ключей WEP, имеются в Интернете в свободном доступе. Таким образом, хуже WEP может быть только полное отсутствие защиты.

Если ваше оборудование позволяет, используйте алгоритм WPA, а еще лучше WPA2. Если же это невозможно, то ваша беспроводная сеть нуждается в дополнительной защите — для этого лучше всего применять VPN.

WPA, WPA2

На смену WEP пришел метод WPA (Wi-Fi Protected Access, защищенный доступ к Wi-Fi), а также его улучшенная версия WPA2. Подробная спецификация WPA2 описана в стандарте 802.11i. Важнейшее различие между WPA и WPA2 заключается в том, что в этих методах используются разные алгоритмы шифрования: RC4 в WPA, AES в WPA2.

WPA разрабатывался как временное решение, и его рассчитывали применять до того, как будет полностью подготовлен стандарт 802.11i. Однако поскольку существует оборудование, которое работает с WPA, но не работает с WPA2, в обозримом будущем будут использоваться оба метода.

Существенное преимущество WPA заключается в том, что ключ применяется только для инициализации соединения. Когда оно установлено, ключ начинает постоянно изменяться в соответствии с хитроумным алгоритмом. В настоящее время WPA и WPA2 считаются надежными, если в качестве пароля применяется достаточно длинная фраза (то есть ключ, состоящий из нескольких слов и дополнительных символов).

В этой книге будет рассмотрен только вариант WPA/WPA2 с применением *предварительно выданного ключа* (Pre-Shared Key или PSK, иногда также называется «персональный WPA»). В таком случае все пользователи WLAN указывают для входа в сеть один и тот же ключ. При применении еще более надежного варианта, называемого *управляемым ключом*, каждый пользователь имеет собственный ключ. В таком случае управление всеми ключами централизованно осуществляется на сервере.

Не забывайте, что при работе с большинством WLAN-роутеров и точек доступа настраивается *один* метод шифрования. Иными словами, невозможно настроить сеть так, чтобы один компьютер устанавливал соединение с помощью WPA2, а второй — с помощью WEP. Таким образом, стандарт безопасности, с которым может функционировать даже самое старое ваше устройство, определяет, с каким стандартом будет работать вся сеть.

Базовая защита

Независимо от того, какую технику шифрования вы используете, необходимо обеспечить и базовую защиту беспроводной сети.

- Настройки точки доступа обычно можно изменить в браузере. Доступ к Интернету защищен паролем, специфичным для конкретной фирмы, — такой пароль обязательно нужно изменить. Следует также свести к минимуму удаленное обслуживание, а если такое обслуживание необходимо, то устанавливать соединение по LAN, а не по WLAN.
- При работе со многими точками доступа WLAN-доступ предоставляется только с определенного MAC-адреса. MAC-адрес (адрес для управления доступом к среде передачи данных) — это уникальный номер контроллера WLAN. Такая

защита является недостаточной, так как хакер вполне может воспользоваться фальшивым MAC-адресом.

- Выключайте точку доступа, если не пользуетесь ею в данный момент.
- Задавайте как можно более длинные ключи и пароли, которые нельзя взломать методом подбора.

Брандмауэр и VPN

Брандмауэр позволяет целенаправленно ограничивать трафик передаваемых по сети WLAN данных определенными протоколами, сегментами сети и т. д. Кроме того, существует точка зрения, согласно которой WLAN, несмотря на любые меры защиты, не гарантирует полной конфиденциальности данных. Чтобы все же обеспечить надежный обмен информацией по беспроводной сети, шифруйте сами данные, передаваемые по сети. Для этого чаще всего применяется VPN.

Поддержка WLAN в Linux

Инструменты для работы в беспроводной сети и команда iw. Ранее инструменты Linux для работы в беспроводной сети играли важную роль при использовании компонентов WLAN. Инструменты для работы в беспроводной сети состоят из многих команд (`iwconfig`, `iwlist` и т. д.), предназначенных для конфигурации WLAN-адаптера.

В наши дни по-прежнему сложно представить себе повседневную работу с WLAN в Linux без этих инструментов, но концептуально они уже устарели. Несколько лет назад началась общая переработка драйверов Linux для работы с WLAN, и была разработана новая управляющая команда `iw`. Но `iw` в любом случае может использоваться только с такими WLAN-контроллерами, чьи драйверы поддерживают интерфейс nl80211.

Поэтому в большинстве дистрибутивов Linux команды `iwconfig` и `iw` устанавливаются параллельно. Базовая информация по системам для работы с беспроводными сетями в Linux изложена на сайтах http://www.hpl.hp.com/personal/Jean_Tourrilhes/Linux/Tools.html и http://linuxwireless.org/.

Аппаратные драйверы. Сами аппаратные драйверы для работы с беспроводными сетями находятся в модулях ядра. Современные драйверы зачастую построены на основе фреймворка mac80211 и, таким образом, совместимы с интерфейсом nl80211 и командой `iw`. В настоящее время в Linux уже есть драйверы практически для всех присутствующих на рынке адаптеров WLAN — но, как обычно, не обходится без отдельных исключений. Особенно сложно обстоит ситуация с наиболее новыми адаптерами WLAN: даже при хорошем взаимодействии между производителями оборудования и сообществом разработчиков Linux может уйти целый год, пока новые драйверы окажутся в конкретном дистрибутиве. Поэтому перед покупкой ноутбука не помешает почитать о нем в Интернете такую информацию.

Встроенное программное обеспечение (прошивка). Большинство контроллеров WLAN являются программируемыми. Чтобы они работали, в ходе инициализации в контроллер должен быть передана так называемая прошивка (программный

код, предназначенный для использования внутри контроллера). Этот код пишется производителями контроллеров и при условии соблюдения необходимых лицензионных условий может распространяться свободно. За передачу кода в контроллер обычно отвечает модуль ядра или система udev. Код контроллера находится в двоичных файлах (блобах, Blobs), которые, в свою очередь, чаще всего располагаются в каталоге `/lib/firmware`.

Производители микросхем обычно предоставляют файлы прошивки в виде двоичного, а не исходного кода. С точки зрения сообщества, выступающего за свободное распространение ПО, это недопустимо, поэтому такой метод подвергается критике, особенно со стороны разработчиков Debian. Мне эта проблема не кажется столь драматичной: ранее в контроллер WLAN было встроено стираемое программируемое запоминающее устройство (EPROM) и никто не думал возмущаться насчет того, что код этого устройства не является открытым. Используемое в настоящее время решение дешевле и может обновляться. Конечно, было бы очень желательно, чтобы все программы, содержащиеся в контроллере, были доступны в виде исходного кода, но такая мечта кажется несбыточной.

Применение драйвера для Windows. Если драйвера для Linux нет, то почему бы не использовать драйвер для Windows? На первый взгляд, это невозможно, но на практике такую замену вполне можно осуществить: интерфейс для интеграции драйверов WLAN под Windows является относительно компактным. В различных коммерческих и свободно распространяемых проектах этот интерфейс (NDIS) был приспособлен для работы с Linux. Вопросы применения в Linux драйверов, предназначенных для Windows, рассматриваются на следующих сайтах:

- http://www.linuxant.com/driverloader/wlan/ — коммерческие;
- http://sourceforge.net/projects/ndiswrapper/ — с открытым кодом.

Разумеется, использование двоичных драйверов противоречит идеям свободного ПО, так как код таких драйверов не предоставляется в распоряжение сообщества разработчиков. Еще один недостаток заключается в том, что существующие драйверы Windows могут работать только в системах, совместимых с Intel/AMD. Но на практике при повседневной работе с Linux драйверы Linux уже не играют практически никакой роли, поскольку для большинства распространенных адаптеров существуют свободные драйверы Linux.

18.3. Активация контроллеров LAN и WLAN вручную

Как правило, при запуске компьютера контроллеры LAN и WLAN автоматически распознаются и инициализируются. В этом разделе рассмотрено, как именно протекает данный процесс и что нужно делать, чтобы при необходимости активизировать эти контроллеры вручную. Разумеется, я не ставлю своей целью научить вас, как настраивать целые сетевые интерфейсы вручную. Гораздо важнее, чтобы вы, прочитав раздел, поняли основы этого процесса. Кроме того, сообщаемая здесь информация должна пригодиться, если у вас возникнут проблемы с активизацией или автоматическим конфигурированием контроллера WLAN.

Активизация контроллера LAN

Обычно сетевой контроллер или контроллер LAN — это микросхема, стоящая на материнской плате вашего компьютера и предоставляющая функции Ethernet. Контроллер может находиться и на отдельной сетевой карте, например, если необходимо оборудовать компьютер дополнительным сетевым интерфейсом. В дальнейшем мы абстрагируемся от того, как физически реализуются сетевые функции, и поговорим только о сетевом контроллере.

Распознавание оборудования. Сначала необходимо удостовериться, что в ядро загружен модуль, необходимый для конкретного сетевого контроллера. Часто ядро делает это автоматически. В таком случае команда выполняется без ошибок (см. ниже). Если на данном этапе возникнут проблемы, необходимо проверить, какой сетевой контроллер установлен в вашем компьютере и какой модуль ядра за него отвечает. Соответствующую информацию в таких случаях выдает команда `lspci`:

```
root# lspci | grep -i net
02:01.0 Ethernet controller: Intel Corporation 82540EP Gigabit Ethernet
                             Controller (Mobile) (rev 03)
...
```

Это означает, что в ноутбуке используется контроллер Gigabit-Ethernet 82540EP от Intel. Далее остается только найти для этого контроллера подходящий драйвер (то есть модуль ядра из каталога `/lib/modules/`*n.n*`/net/*`). Если поискать в Интернете по запросу `linux kernel module 82540EP`, то быстро найдется нужный модуль ядра e1000:

```
root# modinfo e1000
filename: /lib/modules/3.2.0-26-generic/kernel/drivers/net/ethernet/intel/
          e1000/e1000.ko
version: 7.3.21-k8-NAPI
license: GPL
description: Intel(R) PRO/1000 Network Driver
author:      Intel Corporation, <linux.nics@intel.com>
...
```

С помощью команды `lsmod` можно проверить, загрузился ли модуль. Как правило, такое предположение подтверждается — это значит, что Linux правильно распознала контроллер уже при запуске системы. Если же вы получите другой результат, вам понадобится загрузить с помощью `modprobe` подходящий модуль:

```
root# modprobe e1000
```

Команда `dmesg` показывает, не возникло ли при загрузке модуля ошибок (в данном случае ошибок не было). Предупреждение `link is not ready` всего лишь означает, что интерфейс пока не сконфигурирован и поэтому не активен.

```
root# dmesg -c
...
Intel(R) PRO/1000 Network Driver - version 7.3.21-k3-NAPI
Copyright (c) 1999-2012 Intel Corporation.
e1000 0000:02:01.0: PCI INT A -> Link[LNKA] -> GSI 11 (level, low) -> IRQ 11
```

```
e1000: 0000:02:01.0: e1000_probe: (PCI:33MHz:32-bit) 00:11:25:32:4f:5d
e1000: eth0: e1000_probe: Intel(R) PRO/1000 Network Connection
ADDRCONF(NETDEV_UP): eth0: link is not ready
...
```

Как правило, модуль загружается автоматически во время инициализации компьютера. Если это не сработает, то нужно ассоциировать интерфейс `eth0` и модуль ядра `e1000` в конфигурационном файле модуля:

```
# Файл конфигурации модулей /etc/modprobe.conf или /etc/modprobe.d/aliases
alias eth0 e1000
```

Настройка параметров контроллеров. Обычно сетевой контроллер сам распознает параметры, необходимые для обмена данными по сети. Лишь очень редко приходится вручную настраивать такие параметры, как скорость, дуплексный режим и т. д. В подобных ситуациях вам пригодится команда `ethtool` (см. `man ethtool`).

Активизация интерфейса. Активизируйте сетевой интерфейс с помощью команды `ifconfig`:

```
root# ifconfig eth0 up
```

Если модуль ядра не загружен либо загружен неподходящий модуль, выводится сообщение об ошибке `eth0: unknown interface: No such device` (Неизвестный интерфейс. Такого устройства не существует). Более подробную информацию об успешном или неудачном выполнении `ifconfig` дает команда `dmesg`:

```
root# dmesg -c
e1000: eth0: e1000_watchdog: NIC Link is Up 1000 Mbps Full Duplex,
  Flow Control: RX/TX
ADDRCONF(NETDEV_CHANGE): eth0: link becomes ready
eth0: no IPv6 routers present
```

Конфигурирование интерфейса. Чтобы задать конфигурацию сетевого интерфейса, сообщите `ifconfig` его название (`eth0`) и нужный IP-адрес. Если затем повторно выполнить команду, не указывая адреса, то она отобразит всю имеющуюся в системе информацию о сетевом интерфейсе:

```
root# ifconfig eth0 192.168.0.2
root# ifconfig eth0
eth0   Protocol:Ethernet Hardware Address 00:11:25:32:4F:5D
       inet Address:192.168.0.2 Bcast:192.168.0.255 Mask:255.255.255.0
       inet6 Address: fe80::211:25ff:fe32:4f5d/64 scope:connection
       UP BROADCAST RUNNING MULTICAST MTU:1500 Metric:1
       RX packets:82 errors:0 dropped:0 overruns:0 frame:0
       TX packets:49 errors:0 dropped:0 overruns:0 carrier:0
       collisions:0 Send queue size:100
       RX bytes:9252 (9.0 KiB) TX bytes:7732 (7.5 KiB)
       Basic address:0x8000 memory:c0220000-c0240000
```

Теперь с помощью команды `ping` можно проверить, разрешено ли вам связаться с другими компьютерами по сети. Благодаря параметру `-c 2` отсылается ровно два пакета `ping`:

```
root# ping -c 2 192.168.0.1
PING 192.168.0.1 (192.168.0.1) 56(84) bytes of data.
64 bytes from 192.168.0.1: icmp_seq=1 ttl=64 time=2.95 ms
64 bytes from 192.168.0.1: icmp_seq=2 ttl=64 time=0.169 ms

--- 192.168.0.1 ping statistics ---
2 packets transmitted, 2 received, 0% packet loss, time 1002ms
rtt min/avg/max/mdev = 0.169/1.560/2.952/1.392 ms
```

Конфигурация сервера имен. Пока команда `ping` исправно функционирует лишь при условии, что вы правильно укажете IP-адрес. Чтобы можно было находить компьютер и по названию, в файле `/etc/resolv.conf` должен содержаться IP-адрес сервера имен. В следующем примере предполагается, что в локальной сети есть отдельный сервер имен с IP-адресом 192.168.0.1. Но сервер имен может быть и за пределами сети, если его предоставляет интернет-провайдер.

```
# /etc/resolv.conv
nameserver 192.168.0.1
```

Шлюз, задаваемый по умолчанию. Пока вы можете обмениваться пакетами только в пределах локальной сети. Чтобы обеспечить также выход в Интернет, компьютер должен знать, куда направлять такие пакеты. Для этого вам следует указать адрес интернет-шлюза вашей сети с помощью команды `route`. В следующем примере IP-адрес шлюза — 192.168.0.1:

```
root# route add default gw 192.168.0.1
root# route -n
Kernel IP routing table
Destination  Gateway      Genmask       Flags  Metric  Ref  Use  Iface
192.168.0.0  0.0.0.0      255.255.255.0 U      0       0    0    eth0
0.0.0.0      192.168.0.1  0.0.0.0       UG     0       0    0    eth0
```

Теперь вы сможете посылать пакеты по любым адресам в Интернете:

```
root# ping -c 2 yahoo.com
PING yahoo.com (216.109.112.135) 56(84) bytes of data.
64 bytes from w2.rc.vip.dcn.yahoo.com (216.109.112.135): icmp_seq=1 ttl=52
time=116 ms
64 bytes from w2.rc.vip.dcn.yahoo.com (216.109.112.135): icmp_seq=2 ttl=52
time=115 ms

--- yahoo.com ping statistics ---
2 packets transmitted, 2 received, 0% packet loss, time 999ms
rtt min/avg/max/mdev = 115.397/115.807/116.217/0.410 ms
```

Получение информации о DHCP. Если в сети есть DHCP-сервер, вы можете воспользоваться им при конфигурации. После активизации интерфейса (указывается `ifconfig eth0 up` без дополнительных данных), выполните в Debian и Ubuntu команду `dhclient3`:

```
root# dhclient3 eth0
...
Listening on LPF/eth0/00:11:25:32:4f:5d
```

```
Sending on LPF/eth0/00:11:25:32:4f:5d
Sending on Socket/fallback
DHCPDISCOVER on eth0 to 255.255.255.255 port 67 interval 3
DHCPOFFER from 192.168.0.1
DHCPREQUEST on eth0 to 255.255.255.255 port 67
DHCPACK from 192.168.0.1
bound to 192.168.0.15 -- renewal in 36624 seconds.
```

В SUSE для достижения той же цели требуется команда `dhcpcd`:

```
root# dhcpcd eth0
```

Деактивация интерфейса. Чтобы снова деактивировать интерфейс, выполните `ifconfig` с параметром `down`:

```
root# ifconfig eth0 down
```

Управление несколькими контроллерами. Во многих компьютерах содержится по нескольку сетевых контроллеров. Система udev обеспечивает правильное соотнесение оборудования (то есть контроллеров) и названий интерфейсов (`eth0`, `eth1` и т. д.), так что каждый контроллер всегда связывается с одним и тем же интерфейсом (см. подраздел «Взаимное соотнесение контроллеров и сетевых интерфейсов» раздела 18.4).

Активизация контроллера WLAN

Распознавание оборудования. Как правило, модуль ядра для контроллера WLAN загружается автоматически. В большинстве дистрибутивов WLAN-интерфейс называется `wlan0`, но также употребляются имена интерфейсов `eth1` или `ath0` (для контроллера Atheros). Если вы не уверены в том, какие названия употребляются, выведите командой `ifconfig -a` список всех интерфейсов. Команда `dmesg | egrep -i 'wlan|wifi'` выдает все важные сообщения ядра.

Выяснение текущего статуса. С помощью команд `iw dev` *имя* `info` и `iw dev` *имя* `link` можно узнать статус интерфейса. В следующем примере существует активное соединение:

```
root# iw dev wlan0 info
Interface wlan0
   ifindex 3
   type managed
   wiphy 2
root# iw dev wlan0 link
Connected to 00:16:b6:9d:ff:4b (on wlan0)
   SSID: wlan-sol2
   freq: 2462
   RX: 102262 bytes (1784 packets)
   TX: 12815 bytes (86 packets)
   signal: -59 dBm
   tx bitrate: 54.0 MBit/s
   bss flags:      short-preamble short-slot-time
   dtim period:    0
   beacon int:     100
```

Общая информация о фактическом состоянии соединения WLAN также содержится в псевдофайле /proc/net/wireless:

```
root# cat /proc/net/wireless
Inter-| sta-| Quality          | Discarded packets          | Missed | WE
face  | tus | link level noise | nwid crypt frag retry misc | beacon | 22
eth1: 0020 91. 189. 0. 0 0 0 0 4 0
```

Создание и удаление WLAN-интерфейса. Если команда iw не знает интерфейса wlan0 (то есть вы получаете сообщение об ошибке no such device (устройство отсутствует)), это означает, что интерфейс сначала нужно создать. Для этого применяется iw-команда interface add:

```
root# iw phy phy0 interface add wlan0 type managed
```

phy0 адресует первый (а на большинстве компьютеров — единственный) WLAN-контроллер. Вместо managed можно задавать интерфейсы следующих типов: monitor, wds, mesh или mp, а также ibss или adhoc. Новый интерфейс можно дополнительно активизировать.

```
root# ifconfig wlan0 up
```

Как только интерфейс будет вам больше не нужен, его можно удалить с помощью iw del:

```
root# iw dev wlan0 del
```

Установка WLAN-соединения. Команда iw scan выдает подробную информацию обо всех беспроводных сетях, находящихся в пределах досягаемости. Grep выстраивает из этой информации простой список всех сетевых имен (SSID):

```
root# iw dev wlan0 scan | grep SSID
   SSID: myhome
   SSID: wlan-sol2
   ...
```

В незащищенной сети можно установить соединение вручную с помощью iw ... connect. При этом нужно указать имя сети (то есть SSID). Iw отвечает только за установление WLAN-соединения. Ethernet-конфигурация дополнительно осуществляется с помощью dhclient (в SUSE эта команда называется dhcpcd).

```
root# iw dev wlan0 connect hotel-wlan-1
root# dhclient wlan0
```

Команда iw disconnect завершает соединение:

```
root# iw dev wlan0 disconnect
```

WLAN-соединение с помощью WEP. Если для защиты беспроводной сети используется система шифрования WEP, то можно сообщить ключ с дополнительным параметром в форме *n:xxx*, где *n* — номер интерфейса (от 0 до 3). Сам ключ состоит из 5 или из 13 ASCII-символов, либо из 10 или 26 шестнадцатеричных цифр.

```
root# iw dev wlan0 connect hotel-wlan-1 keys 0:1a790bcc31
root# dhclient wlan0
```

WPA-соединение с помощью WPA. Работа WPA или WPA2 построена несколько сложнее. В данном случае в ходе инициализации соединения и дальнейшего обмена информацией применяются постоянно изменяющиеся ключи, генерируемые фоновой программой `wpa_supplicant` из одноименного пакета. Установив этот пакет, создайте конфигурационный файл с названием `/etc/wpa_supplicant.conf`.

В данном файле содержатся некоторые глобальные настройки, а также специфические параметры для конкретных сетей WLAN. В следующем примере показан простейший вариант конфигурации, которого достаточно для соединения компьютера с точкой доступа или WLAN-роутером методом персонального шифрования WPA или WPA2. Два самых важных параметра — это `ssid` для идентификации сети и `psk`, который содержит ключ, повторно зашифрованный в целях безопасности (ключ WPA также можно указать в кавычках обычным текстом).

```
# /etc/wpa_supplicant.conf
network={
   ssid="sol"
   psk=00a38f42e6681596e1a5a4c5ede9a15250fb2a01c21028c6d490bb3458b8ea00
}
network={
   ssid="wlan-sol2"
   psk=053633deb59038da9e9168e015fef97d3d54ae3794d4a12d31ee75a830cccec2
}
```

При шифровании WPA-пароля, состоящего из нескольких слов, поможет команда `wpa_passphrase`. Результат ее выполнения можно скопировать прямо в файл `wpa_supplicant.conf`, после чего следует по возможности удалить ту строку, в которой пароль записан обычным текстом.

```
root# wpa_passphrase sol 'Мой секретный-пресекретный пароль!'
network={
    ssid="sol"
    #psk="Мой секретный-пресекретный пароль!"
    psk=020d93e2ddb2cdee51e800b977ff7d58fde47d0913cd394f2133648a147f513f
}
```

Теперь можно запустить `wpa_supplicant`. Команда будет работать, пока вы не завершите ее, нажав Ctrl+C. Эта команда отвечает за инициализацию WLAN-соединения, а в дальнейшем — за регулярное обновление ключа к соединению. Другими словами, пока вы работаете с WLAN, должна функционировать и эта программа. Продолжайте работу в другой консоли.

Еще несколько замечаний о параметрах команды: `-i` указывает сетевой интерфейс, `-c` — конфигурационный файл (его название вы можете выбрать на свое усмотрение); `-D` указывает, какой драйвер WLAN вы используете. Для начала попробуйте применить wext — это общий интерфейс WLAN, поддерживаемый многими драйверами. Только если этот интерфейс не заработает, прямо укажите свой драйвер, например `-D madwifi`. Справка `man wpa_supplicant` выдает список всех поддерживаемых драйверов:

```
root# wpa_supplicant -i eth1 -D wext -c /etc/wpa_supplicant.conf
Trying to associate with 00:13:46:b5:25:6e (SSID='sol' freq=0 MHz)
```

```
Associated with 00:13:46:b5:25:6e
WPA: Key negotiation completed with 00:13:46:b5:25:6e [PTK=TKIP GTK=TKIP]
CTRL-EVENT-CONNECTED - Connection to 00:13:46:b5:25:6e completed (auth) [id=0 id_str=]
...
```

Подробная документация по `wpasupplicant` имеется на сайте http://w1.fi/wpa_supplicant/.

18.4. Конфигурационные файлы LAN

В этом разделе рассмотрены самые важные конфигурационные файлы, необходимые для подключения компьютера к локальной сети. К сожалению, не для всех этих файлов существуют унифицированные правила, которые использовались бы во всех дистрибутивах. Что касается остальных файлов, то в этом разделе они будут описаны только для дистрибутивов Fedora 12, openSUSE 11.2 и Ubuntu 9.10. Как правило, не следует вносить изменения непосредственно в конфигурационные файлы — лучше использовать инструменты, предусмотренные в вашем дистрибутиве для изменения конфигурации. Рассмотренные в этом разделе конфигурационные файлы, специфичные для отдельных дистрибутивов, важны при работе только в том случае, если вы не используете программу Network Manager!

Для всех примеров, приведенных в данном разделе, действуют следующие правила: конфигурируемый компьютер называется uranus, а его домен — sol. Другие компьютеры в локальной сети называются jupiter, saturn и т. д. В локальной сети используются адреса 192.168.0.*. IP-адрес локального компьютера — 192.168.0.2. Компьютер-шлюз этой локальной сети имеет IP-адрес 192.168.0.1. На компьютере-шлюзе действует собственный сервер имен. Разумеется, названия и адреса приведены только для примера.

Базовая конфигурация

Файл /etc/hosts

В `/etc/hosts` содержится список известных IP-адресов и соответствующих им имен. В этом файле обязательно должны храниться данные о петлевом интерфейсе. Соответствующая запись, как правило, выглядит так:

```
# /etc/hosts
127.0.0.1 localhost       # Петлевой интерфейс компьютера
...
```

В большинстве дистрибутивов Linux вместо 127.0.0.1 допускается запись `::1` (по правилам IPv6). В Red Hat и Fedora в строке `localhost` содержится дополнительная запись `localhost.localdomain`. Эту строку не следует изменять:

```
# /etc/hosts в Red Hat и Fedora
::1 l     localhost.localdomain localhost
...
```

В некоторых дистрибутивах в hosts также содержится запись о локальном компьютере. Если установленный на нем сетевой контроллер конфигурирован с применением статического IP-адреса, то этот адрес необходимо указать. Если же, напротив, сетевой контроллер динамически получает адрес, обращаясь к DHCP, в /etc/hosts указывается псевдоадрес 127.0.1.1 (например, в Debian, Ubuntu) либо такая запись отсутствует (например, в Red Hat, Fedora).

```
# /etc/hosts (Продолжение)
...
192.168.0.2 uranus.sol uranus   # Статический IP-адрес локального компьютера
```

Если вы по имени запрашиваете другой компьютер, расположенный в локальной сети и в этой сети нет сервера имен (см. раздел 19.4), то его название также необходимо указать в /etc/hosts. Вместо ping 192.168.0.13 вы в данном случае можете выполнить просто ping saturn, чтобы проверить связь с компьютером saturn.

```
# /etc/hosts (Продолжение)
...
192.168.0.1 mars.sol mars       # IP-адреса и имена других
192.168.0.2 uranus.sol uranus   # компьютеров в локальной сети
192.168.0.3 saturn.sol saturn
```

Аналогичные записи должны быть в файлах /etc/hosts всех компьютеров, находящихся в локальной сети. Если речь идет об администрировании очень большого количества компьютеров, то управление файлами /etc/hosts значительно усложняется. Поэтому в больших сетях рекомендуется создавать на одном из компьютеров сервер имен. Этому компьютеру (то есть серверу имен) известно, как называются все остальные компьютеры в сети. Компьютеры, находящиеся в локальной сети, могут обращаться к серверу имен, чтобы узнать эту информацию. Теперь /etc/hosts можно сократить до localhost. В любом случае требуется, чтобы файл /etc/resolv.conf был сконфигурирован правильно.

Файл /etc/host.conf

В файле /etc/host.conf указано, как протокол TCP/IP должен узнавать неизвестные IP-адреса. В следующем примере определяется, что сначала интерпретируется файл /etc/hosts (ключевое слово hosts), а затем посылается запрос на сервер имен, указанный в /etc/resolv.conf (bind). Вторая строка позволяет присвоить каждому отдельному хост-имени, заданному в /etc/hosts, несколько IP-адресов.

Этот файл почти во всех дистрибутивах построен именно так, как показано ниже, и изменять его нельзя.

```
# /etc/host.conf
order hosts, bind
multi on
```

Файл /etc/resolv.conf

Файл /etc/resolv.conf управляет тем, как определяются IP-адреса для неизвестных сетевых имен (хост-имен). «Неизвестное» означает, что названия определяются не в hosts.conf.

Применяя ключевые слова domain и search, вы можете дополнять обычные (неполные) названия компьютеров (например, jupiter) доменным именем (до jupiter.sol). Это облегчает работу, так как вы можете указывать локальные хост-имена в сокращенной форме. Для search можно задать несколько доменных имен (для domain — только одно); поэтому имя domain имеет приоритет над search и проверяется в первую очередь. Если здесь будет указано лишь одно доменное имя, то строку domain можно опустить.

Важнейшие записи в /etc/resolv.conf вводятся ключевым словом nameserver. Таким образом, вы можете указать до трех IP-адресов серверов имен. Эти серверы каждый раз запрашиваются, когда необходимо узнать IP-адрес компьютера с неизвестным именем (например, www.yahoo.com). В данном случае нужно обязательно указывать сервер имен, чтобы интернет-адреса можно было преобразовать в IP-адреса. (На большинстве ADSL-маршрутизаторов работает локальная служба DNS, которая, в свою очередь, обращается к DNS провайдера. В крупных локальных сетях обычно функционирует по нескольку серверов имен.)

```
# /etc/resolv.conf
domain sol               # Хост-имена для домена .sol
search sol               # Хост-имена для домена .sol
nameserver 192.92.138.35 # Первая DNS
nameserver 195.3.96.67   # Вторая DNS (на случай отказа первой)
```

Файл resolv.conf создается динамически и зависит от конфигурации конкретной сети:

- если соединение с Интернетом устанавливается с помощью PPP (через модем, ISDN, ADSL, UMTS, VPN), то сценарий, устанавливающий соединение, автоматически вносит адреса nameserver вашего интернет-провайдера в /etc/resolv.conv;
- если ваше соединение по локальной сети (LAN, WLAN) конфигурируется с применением DHCP, то сценарий, который устанавливает соединение, записывает адреса nameserver, переданные с DHCP-сервера;
- Ubuntu в версии 12.04 и выше по умолчанию создает локальный сервер имен. При этом применяется программа dnsmasq, с которой вы познакомитесь в следующей главе. Таким образом, сервер имен имеет адрес 127.0.0.1, то есть указывает на localhost.

  Если вы хотите создать сервер имен вручную, то нужно указать его адрес /etc/network/interfaces с ключевым словом dns-nameservers:

```
# Файл /etc/network/interfaces (Ubuntu 12.04 и выше)
...
auto eth0
   iface eth0 inet static
   ...
   dns-nameservers 10.0.17.1
```

За применение /etc/resolv.conf отвечает пакет resolvconf. Непосредственное изменение /etc/resolv.conf вручную не предусмотрено!

Защита resolv.conf от изменений

В большинстве случаев целесообразно применять автоматическую настройку файла `resolv.conf`. Однако если вам это не нужно, то автоматическое изменение в большинстве случаев можно отменить.

При применении в Debian и Ubuntu PPP-соединений нужно удалить ключевое слово `usepeerdns` из `/etc/ppp/peers/`*имя*. Что касается сетевых интерфейсов DHCP, ситуация зависит от того, какой DHCP-клиент у вас установлен. Если установлен dhcp3-клиент, вам необходимо изменить его конфигурационный файл `dhclient.conf`:

```
# /etc/dhcp3/dhclient.conf
...
supersede domain-name "sol";
prepend domain-name-servers 192.168.0.1;
```

В Red Hat и Fedora в таком случае необходимо изменить файл `ifcfg-xxx` для соответствующего интерфейса:

```
# /etc/sysconfig/network-scripts/ifcfg-xxxx (Red Hat, Fedora)
PEERDNS=no
```

В SUSE измените следующий конфигурационный файл:

```
# /etc/sysconfig/network/config (SUSE)
NETCONFIG_DNS_POLICY=""
```

Конфигурация шлюза

Для этой операции не существует единого стандарта, например не указывается, в каком файле выполняется конфигурация шлюза. В локальных сетях адрес шлюза обычно определяется через DHCP. При статической конфигурации эта задача решается в различных файлах, в зависимости от дистрибутива.

В Debian и Ubuntu в файле `/etc/network/interfaces` описаны все сетевые интерфейсы. Если конфигурация интерфейса является статической, то шлюз настраивается с помощью ключевого слова `gateway`:

```
# в /etc/network/interfaces (Debian, Ubuntu)
...
iface eth0 inet static
  address 192.168.0.2
  netmask 255.255.255.0
  gateway 192.168.0.1
```

В Red Hat и Fedora в конфигурационном файле содержится переменная `GATEWAY`, отвечающая за сетевой интерфейс:

```
# /etc/sysconfig/network-scripts/ifcfg-xxxx (Red Hat, Fedora)
GATEWAY=192.168.0.1
```

В SUSE конфигурация централизованно осуществляется в следующем файле:

```
# in /etc/sysconfig/network/routes (SUSE)
default 192.168.0.1 - -
```

Фактическая настройка шлюза во всех дистрибутивах производится одинаково: с помощью команды `route`. На примере двух следующих команд показано, как вручную добавить в таблицу маршрутизации шлюз 192.168.0.1, а затем снова удалить его из таблицы:

```
root# route add default gw 192.168.0.1
root# route del default gw 192.168.0.1
```

Конфигурация хост-имен

Актуальное хост-имя можно узнать с помощью команды `hostname`. Если хост-имя настраивается без применения DHCP, то конфигурация в Debian и Ubuntu производится в файле `/etc/hostname`, а в SUSE — в файле `/etc/HOSTNAME`. В Red Hat и Fedora хост-имя настраивается с помощью одноименной переменной (`HOSTNAME`) в файле `/etc/sysconfig/network`. Не забудьте, что еще нужно настроить `/etc/hosts`, если в этом файле есть строка с хост-именем компьютера.

Взаимное соотнесение контроллеров и сетевых интерфейсов

При использовании сетевых интерфейсов обычно бывает сложно определить взаимосвязи между устройствами `eth`*n* и физическим оборудованием. При работе с некоторыми устройствами можно применить команду `ethtool -p eth0 10`, которая заставляет в течение 10 секунд мерцать светодиод, встроенный в гнездо того или иного устройства. Если сетевой драйвер не поддерживает такую операцию, вы получите сообщение об ошибке `Operation not supported` (Операция не поддерживается).

В большинстве современных дистрибутивов Linux за соотнесение сетевых контроллеров и интерфейсов отвечает система udev. Если точнее, то названиями сетевых интерфейсов управляет файл `net_persistent_names.rules`. Этот файл может выглядеть, например, так:

```
# Файл /etc/udev/rules.d/70-persistent-net.rules
SUBSYSTEM=="net", ACTION=="add", DRIVERS=="?*", ATTR{address}=="00:16:17:cd:c3:81",
ATTR{type}=="1", KERNEL=="eth*", NAME="eth0"
SUBSYSTEM=="net", ACTION=="add", DRIVERS=="?*", ATTR{address}=="00:14:6c:8e:d9:71",
ATTR{type}=="1", KERNEL=="eth*", NAME="eth1"
SUBSYSTEM=="net", ACTION=="add", DRIVERS=="?*", ATTR{address}=="00:4f:4e:0f:8e:a0",
ATTR{type}=="1", KERNEL=="eth*", NAME="eth2"
```

В Fedora этот процесс организован иначе: сетевые интерфейсы, интегрированные в материнскую плату, получают имена вида `em`*n*, причем их нумерация начинается с 1. Сетевые устройства, подключенные через шину PCI, получают обозначения `psp`*n*, где `s` означает PCI-ячейку, а *n* — номер порта (нумерация также начинается с 1). На моем тестовом компьютере со встроенным сетевым интерфейсом и вторым PCI Ethernet-адаптером интерфейсы получили имена `em1` и `p37p1`, а на виртуальной машине — `p2p1`. В документации Fedora указано, что устройства на виртуальной машине должны называться по образцу `eth`*n*, но по крайней мере для

VirtualBox это не соблюдается. Имена WLAN-интерфейсов остаются неизменными (то есть `wlann`). Более подробно о новой системе именования сетевых интерфейсов рассказано на сайтах http://domsch.com/blog/?p=455 и http://fedoraproject.org/wiki/Features/ConsistentNetworkDeviceNaming.

18.5. Конфигурация сети вручную

В этом разделе мы поговорим о том, как конфигурация сети осуществляется вручную, без помощи различных конфигурационных инструментов. Описанные здесь методы целесообразны, в частности, в тех случаях, когда вы настраиваете сервер или виртуальную машину. В обоих случаях обычно приходится обходиться как без графического пользовательского интерфейса, так и без соответствующих инструментов администрирования. Вам придется самостоятельно редактировать специфичные для конкретных дистрибутивов файлы, отвечающие за конфигурацию сети.

Разумеется, нижеизложенные рекомендации применимы и к настольным системам. Но перед тем как приступать к подобной работе, нужно обязательно деинсталлировать сетевой менеджер!

Fedora и Red Hat

Удаление сетевого менеджера. В системе Fedora или Red Hat на настольном компьютере сначала удаляем сетевой менеджер:

```
root# yum remove NetworkManager
```

DHCP-конфигурация. В Fedora и Red Hat конфигурация любого сетевого интерфейса происходит в файле `DHCP/etc/sysconfig/network-scripts/icfg-xxx`, где `xxx` — имя сетевого интерфейса. В следующих примерах я использую `eth0`, но вы должны указать интерфейс в соответствии с вашей системой. Как правило, конфигурационные файлы уже существуют, и вам понадобится просто изменить в них настройки.

Если будет необходимо узнать MAC-адрес сетевого адаптера, воспользуйтесь командой `ifconfig -a`.

Если компьютер получает сетевые параметры по DHCP, то просто создайте файл `ifcfg-eth0` следующим образом:

```
# Файл /etc/sysconfig/network-scripts/ifcfg-eth0
DEVICE=eth0
HWADDR=xx:xx:xx:xx:xx:xx (собственный MAC-адрес)
NM_CONTROLLED=no
ONBOOT=yes
BOOTPROTO=dhcp
TYPE=Ethernet
USERCTL=no
PEERDNS=yes
IPV6INIT=no
```

Статическая конфигурация. При статической конфигурации файл должен строиться по следующему образцу, причем значения IP-адресов и масок нужно заменить собственными значениями:

```
# Файл /etc/sysconfig/network-scripts/ifcfg-eth0
DEVICE=eth0
HWADDR=xx:xx:xx:xx:xx:xx (собственный MAC-адрес)
NM_CONTROLLED=no
ONBOOT=yes
BOOTPROTO=none
TYPE=Ethernet
USERCTL=no
IPV6INIT=no
IPADDR=10.0.17.33
NETMASK=255.255.255.0
NETWORK=10.0.17.0
BROADCAST=10.0.17.255
GATEWAY=10.0.17.1
```

Адрес шлюза можно настраивать не только в `ifcfg-xxx`, но и в `/etc/sysconfig/network`. Это целесообразно в тех случаях, когда для всех сетевых интерфейсов существует центральный шлюз. Сервер имен (или серверы имен) записываются в файле `/etc/resolv.conf`:

```
# /etc/resolv.conf
nameserver 10.0.17.1  # Первый DNS-сервер
nameserver 10.0.17.2  # Второй DNS-сервер
```

Хост-имя. Если вы хотите изменить хост-имя, то найдите соответствующий параметр в файле `/etc/sysconfig/network`. Как правило, бывает целесообразно ассоциировать IP-адрес компьютера и его имя и в файле `/etc/hosts`:

```
# /etc/hosts
...
10.0.17.33 myhostname.mydomainname myhostname
```

Активизация конфигурации. Наконец, вы запускаете Init-сценарий, отвечающий за ручную конфигурацию сети:

```
root# service network start (Действует сразу)
root# chkconfig network on  (Действует после перезапуска компьютера)
```

Debian и Ubuntu

Удаление сетевого менеджера. В Debian и Ubuntu также для начала нужно удалить сетевой менеджер (если он установлен):

```
root# apt-get remove network-manager
```

/etc/network/interfaces. За конфигурацию всех интерфейсов отвечает файл `/etc/network/interfaces`. Его синтаксис исключительно прост: любой интерфейс, который необходимо активизировать при запуске компьютера, именуется в этом

файле командой `auto` *имя*. Команда `iface` *имя параметры* описывает базовую конфигурацию интерфейса.

При статической конфигурации в следующих строках идут параметры `address` (IP-адрес), `netmask` (маска допустимого адресного пространства) и `gateway` (IP-адрес шлюза, соединяющего данный компьютер с Интернетом либо с другим компьютером).

DHCP-конфигурация. Если компьютер получает сетевые параметры через DHCP, то вся конфигурация занимает всего четыре строки! Первые две строки активизируют обратный петлевой интерфейс, без которого не обойтись. Этот интерфейс служит для внутреннего обмена информацией в самом компьютере. Две оставшиеся строки активизируют интерфейс `eth0`.

```
# /etc/network/interfaces
auto lo
iface lo inet loopback

# Динамическое подключение к DHCP-серверу,
# сообщающему ключевые данные о доступе к Интернету
auto eth0
iface eth0 inet dhcp
```

Статическая конфигурация. Если подключение к Интернету сконфигурировано статически, то в строке `iface` содержится ключевое слово `static`. Сетевые параметры задаются последовательно, через ключевые слова, значение которых является очевидным:

```
# /etc/network/interfaces
auto lo
iface lo inet loopback

# Статическая конфигурация сети
auto eth0
iface eth0 inet static
   address      211.212.213.37
   netmask      255.255.255.224
   gateway      211.212.213.1
```

В современных версиях Ubuntu (12.04 и выше) можно также задавать адреса одного или нескольких DNS-серверов. За интерпретацию ключевого слова `dns-nameservers` отвечает пакет `resolvconf`.

```
# /etc/network/interfaces (Ubuntu 12.04 и выше)
...
auto eth0
iface eth0 inet static
   ...
   dns-nameservers 211.222.233.244 212.223.234.245
```

Напротив, в сравнительно старых версиях Ubuntu, а также в Debian нужно указывать адреса серверов имен непосредственно в `/etc/resolv.conf`:

```
# /etc/resolv.conf (Debian, Ubuntu до версии 11.10)
nameserver 211.222.233.244      # Первый DNS-сервер
nameserver 212.223.234.245      # Второй DNS-сервер
```

Хост-имя. Если вы хотите задать хост-имена заново, то внесите изменения в файлы `/etc/hostname` и `/etc/hosts`.

Активизация изменений. Чтобы изменения конфигурации вступили в силу, выполните следующую команду:

```
root# /etc/init.d/networking restart
```

В актуальных версиях Debian и Ubuntu данная команда выдает предупреждение `/etc/init.d/networking restart is deprecated` (перезапуск сети устарел). Но, как правило, эта команда функционирует. Достойной альтернативы для этой команды пока не существует, подробнее об этом можно почитать на сайтах http://bugs.debian.org/cgi-bin/bugreport.cgi?bug=565187 и http://tinyurl.com/6uqdbpt.

Сами конфигурационные задачи выполняются специфичными для Debian и Ubuntu командами `ifup` и `ifdown` из пакета `ifupdown`. Команда `ifup -a` интерпретирует файл `/etc/network/interfaces` и активизирует все `auto`-интерфейсы. Поскольку интерфейсы конфигурируются через DHCP, команда `ifup` обращается к `dhclient` для передачи и интерпретации DHCP-данных. За конфигурацию отвечает файл `/etc/dhcp3/dhclient.conf`.

Не прибегая к `networking restart`, можно активизировать измененную конфигурацию для сетевого интерфейса:

```
root# ifdown eth0
root# ifup eth0
```

SUSE

При работе с дистрибутивами SUSE рекомендуется применять для конфигурации YaST. При минимально необходимой установке YaST также предоставляется в текстовом режиме. В модуле Сетевые устройства ▸ Сетевые настройки установите флажок Традиционные методы с применением ifup и деактивируйте, таким образом, сетевой менеджер. Затем перейдите на вкладку Обзор и откорректируйте там настройки сетевого адаптера.

18.6. Zeroconf и Avahi

В этой книге я обычно исхожу из того, что вы либо конфигурируете компьютер в своей сети самостоятельно, либо берете конфигурацию IP с центрального роутера или DHCP-сервера. Но есть еще и третий способ — автоматическая конфигурация с помощью Zeroconf.

При применении этого метода все компьютеры, расположенные в сети, обмениваются своими данными о конфигурации. Новые компьютеры или устройства, подключаемые к сети, сами конфигурируются на основании этой информации так, что могут обмениваться информацией с другими устройствами без возникновения

конфликтов. Компьютеры, которые настраиваются автоматически, используют IP-адреса из диапазона 169.254.*.* и хост-имена, оканчивающиеся на `.local`. Обмен информацией Zeroconf производится через UDP-порт 5454. Чтобы Zeroconf работал, этот порт в пределах сети WLAN не должен блокироваться брандмауэром!

Zeroconf впервые был применен под названием Rendezvous компанией Apple; позже этот проект переименовали в Bonjour, и теперь им также можно пользоваться в Windows. Хотя данное решение и предоставляется в виде открытого кода, оно не соответствует правилам лицензии GPL. Поэтому в Linux развился собственный проект Zeroconf, называемый Avahi, — его код не зависит от Bonjour. Проект Avahi работает по лицензии LGPL. (Откуда взялось странное название Avahi, мне не известно[1].)

Программы, совместимые с Zeroconf, могут отображать все остальные компьютеры из данной сети, которые работают с Zeroconf, а также их ресурсы (например, сетевые каталоги, SSH, серверы HTTP и FTP). Таким образом, не настраивая конфигурацию отдельно, вы можете интегрировать в сеть два и более компьютера и осуществлять обмен данными между ними.

Все более широкое распространение (ADSL)-маршрутизаторов приводит к тому, что Zeroconf, в сущности, становится избыточным. Причина, по которой пакеты с Avahi обычно устанавливаются по умолчанию во многих дистрибутивах, заключается, скорее, в функциях обзора (browsing). Тот факт, что компьютеры видят и «знают друга друга по именам» и эти возможности никак не связаны с конфигурацией сети — просто замечателен. Кроме того, Apple использует программу Bonjour практически на всех своих устройствах (Mac, iPhone, iPad и т. д.). Если вы хотите, чтобы компьютер с Linux был виден с устройств Mac (например, для реализации функции AirPrint), на компьютере должен работать демон Avahi. Более подробная информация и советы содержатся на сайте http://avahi.org.

Демон avahi. За обмен информацией между компьютерами, на которых установлена программа Avahi, отвечает служба `avahi-daemon`. Ее конфигурация производится в файле `/etc/avahi/avahi-daemon.conf`, причем основные настройки обычно можно оставить без изменения. Единственным исключением является переменная `enable-dbus`: она определяет, задействует ли Avahi механизм обмена данными. Для работы некоторых Avahi-совместимых программ требуется DBUS (шина передачи данных). Чтобы активировать DBUS, измените файл `avahi-daemon.conf` следующим образом:

```
# /etc/avahi/avahi-daemon.conf
[server]
...
enable-dbus=yes
```

Затем повторно запустите службу:

```
root# /etc/init.d/avahi-daemon restart
```

---

[1] Авахи — это название древесного лемура, мордочка которого является символом проекта (http://en.wikipedia.org/wiki/Woolly_lemur). — *Примеч. перев.*

Преобразование имен в IP-адреса. Если вы хотите обратиться к внешним компьютерам, на которых работают обычные сетевые программы, не совместимые с Avahi (например, `ping merkur.local`), нужно установить еще один сетевой демон с помощью `avahi-dnsconfd` и запустить его. В данном случае мы имеем дело со своего рода сервером имен для работы с хост-именами Avahi (если в вашей сети и так работает сервер имен, то этот демон вам не потребуется).

Чтобы при преобразовании имен в IP-адреса все программы обращались к `avahi-dnsconfd`, убедитесь, что у вас установлена библиотека `libnss-mdns` и в строке `hosts:` файла `/etc/nsswitch.conf` есть ключевое слово `mdns4`. Как правило, такие настройки используются по умолчанию.

```
# в /etc/nsswitch.conf
...
hosts: files dns mdns4
...
```

Просмотр сети и ресурсов. Завершив эти подготовительные работы, вы можете проверить, о каких компьютерах или службах, находящихся в вашей сети, известно Avahi. Для этого используется команда консоли `avahi-browse -a -t` или ее графический аналог `avahi-discovery` (рис. 18.3). По умолчанию программа Nautilus в режиме просмотра сети отображает все компьютеры, на которых работает Avahi. В Konqueror по адресу `zeroconf:/` расположена схожая функция при условии, что вы установили соответствующее дополнение (пакет зависит от дистрибутива, например `kde-zeroconf`). Мультимедийные приложения и программы для рассылки сообщений также поддерживают Zeroconf.

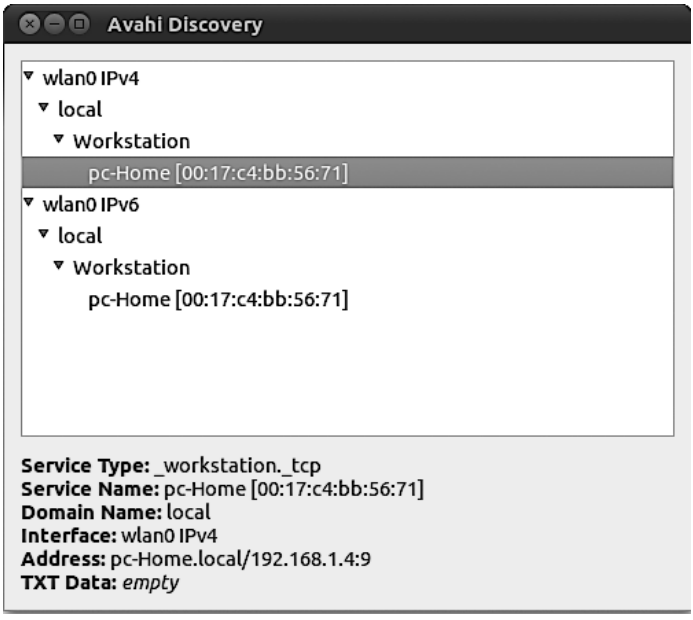


Рис. 18.3. Просмотр ресурсов сети с помощью программы Avahi Discovery

18.7. Основы PPP

Протокол «точка — точка» (Point-to-Point Protocol, PPP) обеспечивает соединение TCP/IP между двумя компьютерами через последовательный интерфейс. Протокол применяется при работе с аналоговыми модемами, а также модемами ISDN, ADSL и UMTS.

В Linux за соединение по протоколу PPP отвечает программа `pppd`. Она может использоваться как на клиентском компьютере, так и на сервере, но мы рассмотрим только работу с клиентом.

В данном разделе делается введение в основы PPP. В следующих разделах мы изучим конкретные способы применения PPP при различных видах доступа к Интернету.

Варианты PPP. Программа `pppd` предназначена только для работы с PPP-соединениями, устанавливаемыми через последовательную телефонную линию, то есть через аналоговый модем. Для работы с ADSL-соединениями применяются три усовершенствованных варианта протокола: PPPoE, PPPoA или PPTP. Рассмотрим их подробнее.

- **PPPoE.** Протокол двухточечного соединения через Ethernet — общедоступный документированный протокол, данные по которому содержатся в RFC 2516. Его основной недостаток заключается в том, что он ограничивает максимальный размер пакетов (проблема MTU).
- **PPPoA.** Протокол туннелирования типа «точка — точка» по ATM — альтернатива PPPoE. Здесь ATM означает Asynchronous Transfer Mode (Асинхронный режим передачи).
- **PPTP.** Туннельный протокол типа «точка — точка» — протокол Microsoft, развитый из других стандартов, документация по которому также имеется в свободном доступе (RFC 2637). Первоначально он предназначался для обеспечения работы виртуальных частных сетей.

Аутентификация. Построение любого PPP-соединения начинается с аутентификации: это означает, что ваш компьютер должен быть опознан провайдером Интернета по логину и соответствующему паролю. Эта операция может осуществляться одним из двух способов: PAP и CHAP. CHAP, в свою очередь, имеет различные варианты (например, MS-CHAPv2), повышающие надежность соединения.

- **PAP.** Применяя протокол аутентификации пароля, клиент (то есть ваш компьютер) обычно передает логин и пароль в незашифрованном виде (также есть вариант PAP и с шифрованием).
- **CHAP.** При использовании протокола аутентификации по квитированию вызова сервер (то есть провайдер) при аутентификации отсылает клиенту так называемый пакет challenge. Программа `pppd` использует эти данные, чтобы рассчитать из пароля хеш-значение. Затем `pppd` отсылает логин и хеш-значение обратно провайдеру (таким образом, необходимость передачи самого пароля пропадает).

Конфигурационные файлы и сценарии pppd

/etc/ppp/options. В /etc/ppp/options содержатся глобальные параметры pppd. Они устанавливаются по умолчанию для всех соединений, создаваемых pppd. Справка по важнейшим параметрам приводится в следующем подразделе.

Если вы используете pppd для создания различных интернет-соединений (например, через аналоговый модем или ADSL-модем), в options должно содержаться как можно меньше настроек. Чтобы конфигурировать параметры, пользуйтесь специальными файлами для работы с соединениями, находящимися в /etc/ppp/peers/! Таким образом вы будете избавлены от проблемы, когда параметр, который подходит для варианта соединения A, может заблокировать соединение, создаваемое по варианту B.

/etc/ppp/peers/name. В /etc/ppp/peers/*имя* содержатся параметры, касающиеся конкретного типа соединения. Чтобы запустить pppd с применением этих параметров, выполните следующую команду:

```
root# pppd call имя
```

Настройки в /etc/ppp/peers/*имя* имеют приоритет перед /etc/ppp/options.

/etc/ppp/pap-secrets и chap-secrets. Эти файлы содержат список всех логинов и паролей для аутентификации PAP и CHAP. Если вы не знаете точно, как осуществляется аутентификация — через PAP или через CHAP, просто вставьте одну и ту же запись и в ap-secrets, и в chap-secrets. Значительно более надежны и поэтому шире распространены варианты CHAP (CHAP, MS-CHAP или MSCHAPv2).

Записи для клиентских соединений выглядят так:

```
#/etc/ppp/pap-secrets и /etc/ppp/chap-secrets
#логин    IP-адрес сервера   Пароль         IP-адрес клиента
"hofer"   *                  "qwe44trE"     *
```

Вместо символа * между логином и паролем можно указать IP-адрес того PPP-сервера, с которым необходимо установить соединение. В таком случае информация о пароле действительна только для данного IP-адреса. Это дополнительный механизм защиты от злонамеренного использования, но применить его можно лишь в том случае, если IP-номер известен и не изменяется (при использовании ADSL-соединений по протоколу PPTP вы можете указать IP-адрес ANT, то есть оконечного сетевого устройства для работы с ADSL).

Вместо второй звездочки вы можете задать, какому клиентскому IP-адресу должна соответствовать данная комбинация логина и пароля. Как правило, этот адрес неизвестен, так как при каждом новом соединении сервер присваивает клиенту новый IP-адрес — свободный в настоящий момент. Поэтому в предыдущем примере звездочка (которую можно опустить) означает, что клиент может иметь любой IP-адрес.

Если pppd применяется не на клиенте, а на сервере, то в pap-secrets и chap-secrets не указываются логины внешних клиентов. Пример конфигурации приведен в подразделе «Настройка PPTPD-сервера» раздела 27.2. Там рассказывается о создании VPN-сервера на базе PPTP.

/etc/ppp/ip-up и /etc/ppp/ip-down. Сценарные файлы /etc/ppp/ip-up и /etc/ppp/ip-down выполняются сразу же после установления либо завершения соединения. Они могут применяться для настройки /etc/resolv.conf или создания и изменения функций роутинга и маскарадинга, а также настроек брандмауэра.

Обоим сценариям сообщается по шесть параметров. Первый параметр содержит название интерфейса (например, ppp0), второй и третий не используются; четвертый включает локальный IP-адрес, пятый — IP-адрес партнера по PPP, шестой — последовательность символов, используемую для идентификации PPP-соединения (параметр ipparam).

Если при установлении соединения передаются адреса DNS (параметр usepeerdns), то оба адреса также предоставляются в переменных DNS1 и DNS2.

СОВЕТ

Обратите внимание на то, что сценарные файлы являются исполняемыми (chmod u+x)! Почти со всеми дистрибутивами поставляются заранее сконфигурированные файлы ip-up и ip-down.

В зависимости от дистрибутива потребуется внести некоторые изменения в каталогах ip-up.local и ip-down.local или в дополнительных сценарных файлах из каталогов ip-up.d и ip-down.d.

Параметры pppd

Программе pppd известны десятки параметров, настройка которых обычно производится в файлах /etc/ppp/options и /etc/ppp/peers/name. Полная справка по параметрам приводится в файле man pppd. Ниже в алфавитном порядке перечислены и кратко описаны только самые важные параметры — и даже их набралось немало.

- connect *команда* — выполняется перед запуском pppd. При работе с аналоговыми модемами в данном случае обычно вызывается команда chat, предназначенная для того, чтобы набрать телефонный номер провайдера. Этот параметр применяется и в тех случаях, когда используется коммутируемое соединение по запросу (параметр demand). При ADSL-соединениях в качестве команды connect может применяться просто /bin/true.
- crtscts — поток данных, идущий через последовательный интерфейс, контролируется RTS/CTS. Параметр crtscts используется только при установлении соединений через модем.
- debug — процесс установления соединения подробно протоколируется посредством программы syslogd (то есть в файлах /var/log/*xxx*, в зависимости от конфигурации syslogd).
- defaultroute — когда PPP-соединение установлено, IP-адрес задается как конечный адрес для пересылаемых пакетов. Этот параметр требуется почти всегда (если вы не настраиваете функции роутинга сами, например с помощью сценария ip-up).
- demand — соединение устанавливается не сразу, а лишь тогда, когда на самом деле необходимо передать данные. В таком случае выполняется сценарий, указанный в connect (demand также автоматически активизирует persist, если вы специально не отмените эту операцию параметром nopersist).

- `idle` *n* — соединение должно автоматически завершаться, если в течение *n* секунд не передаются никакие данные. Эта функция полезна, если вы забыли отключить соединение. Она сэкономит вам немало денег за телефон!
- `ipparam` — с помощью этого параметра можно указать последовательность символов. После установления соединения `pppd` передает данную последовательность символов сценарию `/etc/ppp/ip-up`. В некоторых дистрибутивах эта дополнительная информация используется для выбора нужного файла конфигурации сети (`/etc/sysconfig/network-scripts/ifcfg-`*ipparam*).
- `ktune` — позволяет `pppd` изменять настройки ядра. Такие изменения могут быть нужны при использовании *коммутируемого соединения по запросу*, чтобы пакет данных, инициирующий сетевое соединение, не потерялся.
- `lcp-echo-interval` *n* — каждые *n* секунд посылает провайдеру эхо-запрос. Таким образом проверяется, работает ли еще соединение.
- `lcp-echo-failure` *n* — указывает, после какого количества неудачных запросов `pppd` должна завершить соединение. В зависимости от настроек `persist`/`nopersist`, работа `pppd` может быть завершена либо может быть предпринята новая попытка соединения.
- `linkname` *имя* — благодаря этому параметру в качестве файла идентификации процесса (PID-файла) используется имя `/var/run/ppp-`*имя*`.pid`. Иногда бывает нужно завершить определенный процесс из нескольких, работающих одновременно. В таком случае номер нужного процесса можно с легкостью считать из PID-файла (по умолчанию название PID-файла — `/var/run/pppn.pid`).
- `lock` — `pppd` создает блокирующий файл для интерфейса передачи данных (например, последовательного интерфейса при использовании модема). Таким образом, исключается ситуация, в которой к интерфейсу могут одновременно обратиться две и более программы.
- `mru` *n* и `mtu` *n* — устанавливает желаемые значения для *максимальной длины получаемых пакетов (MRU)* и *максимальной длины передаваемых пакетов (MTU)*. Величины MRU и MTU указывают размер блока пакета с файлами. Обычно для MRU действует настройка, задаваемая по умолчанию и равная `1500`. При работе с PPPoE это значение необходимо снизить до `1492`. Текущее значение можно проверить с помощью `ifconfig` после установления PPP-соединения.
- `name` *«abc"* — если `pppd` используется на клиентском компьютере, то он применяет логин, например *abc* (соответствующий пароль можно узнать из файла PAP/CHAP).

  Если же `pppd` используется на сервере, то `abc` — это имя локальной системы. Оно должно соответствовать данным, указанным в правом столбце файла PAP/CHAP (см. подраздел «Настройка PPTPD-сервера» раздела 27.2).
- `noauth` — удаленная станция `pppd`, то есть провайдер, также называемый в документации по `pppd "peer"` (одноранговый узел) в аутентификации не нуждается. Этот параметр требуется почти всегда. Однако вы сами (как клиент) обязательно должны аутентифицироваться!

- `noaccomp`, `nopcomp`, `novj`, `novjccomp`, `nobsdcomp`, `nodeflate`, `noccp` — деактивируют всевозможные механизмы сжатия. При работе со многими провайдерами необходимо указать как минимум `nopcomp`. Зачастую рекомендуется применять и другие параметры. Если возникают проблемы, не помешает попробовать использовать эти параметры.
- `nodetach` — PPP, в отличие от многих собратьев, запускается не как фоновый процесс. Вся информация об управлении выводится в окно терминала, в котором была запущена программа. Этот параметр хорошо подходит для тестирования системы.
- `noipdefault` — благодаря данному параметру провайдер (а не компьютер) назначает IP-адрес для PPP-соединения. Поскольку почти все интернет-провайдеры распределяют IP-адреса динамически, этот параметр, как правило, должен применяться. При каждом входе в систему вы получаете другой IP-адрес, а точнее первый IP-адрес, свободный в данный момент. Метод аналогичен тому, что применяется при работе с DHCP-сервером в локальной сети — см. раздел 19.5.
- `nopersist` — параметр, обратный `persist`. Программа `pppd` завершает работу, если соединение обрывается (или по истечении срока, заданного параметром `idle`).
- `persist` — при ненамеренном обрыве соединения `pppd` автоматически пытается восстановить соединение. При использовании `demand` этот параметр применяется автоматически.
- `plugin` *`имя`*`.so` — программа `pppd` при запуске загружает PPP-модуль *`имя`*`.so`. Дополнительные модули PPP обычно находятся в каталоге `/usr/lib/pppd/n/`. Параметр задействуется в том случае, если используются специальные PPP-функции, реализованные в виде PPP-плагина (например, протоколы PPPoE, PPPoA и PPTP или специальные механизмы аутентификации и шифрования).
- `pty` *`сценарий`* — указывает сценарий или программу, которые используются для обмена информацией вместо устройства. Параметр необходим, если `pppd` должен использовать внешнюю программу, поддерживающую дополнительный протокол обмена информацией (например, PPPoE, PPPoA или PPTP).
- `refuse-pap`, `-chap`, `-mschap-v2` — благодаря этим параметрам не допускаются указанные методы аутентификации.
- `require-pap`, `-chap`, `-mschap-v2`, `-mppe`, `-mppe-128` — определяют, какой метод аутентификации будет применяться (PAP, CHAPT, Microsoft-CHAP Version 2) и как должны шифроваться данные (MPPE — протокол Microsoft шифрования двухточечного соединения). Если не указать ни один из этих параметров, то `pppd` сама выберет наилучший метод (в зависимости от требования удаленной станции).
- `usepeerdns` — многие PPP-серверы при установлении соединения передают два DNS-адреса. Благодаря параметру `usepeerdns` программа `pppd` узнает эти адреса при установлении соединения и передает сценарию `ip-up`. Во многих дистрибу-

тивах конфигурация этого сценария такова, что адреса DNS автоматически заносятся в файл /etc/resolv.conf.

Некоторые другие параметры описаны в разделе 19.6, где рассказывается о конфигурации PPTP-сервера, обеспечивающего безопасность доступа к WLAN с помощью VPN.

18.8. Внутренняя организация UMTS

Современные модемы чаще всего выглядят как флешки. Хотя для них закрепилось обозначение UMTS, большинство модемов совместимо и с более ранними стандартами (GSM, GPRS, EDGE). Модем, в зависимости от стандарта связи, автоматически находит самую быструю доступную сеть.

Драйвер. UMTS-модемы хорошо поддерживаются в Linux, что не может не радовать. Если вы работаете с современным дистрибутивом, то большинство современных дистрибутивов опознаются системой сразу при подключении. Вы можете в этом убедиться с помощью команды dmesg:

```
root# dmesg
...
usb 1-4: new high speed USB device using ehci_hcd and address 3
...
USB Serial support registered for GSM modem (1-port)
option 1-4:1.0: GSM modem (1-port) converter detected
usb 1-4: GSM modem (1-port) converter now attached to ttyUSB0
option 1-4:1.1: GSM modem (1-port) converter detected
usb 1-4: GSM modem (1-port) converter now attached to ttyUSB1
usbcore: registered new interface driver option
option: v0.7.2:USB Driver for GSM modems
```

Если оборудование удачно определилось, то соединение с Интернетом тоже установится без проблем — используйте Network Manager или UMTSmon (см. разделы 18.1–18.2).

Внутри системы Linux большинство модемов воспринимаются как устройства для последовательной передачи данных. Некоторые устройства требуют для работы драйвер Nozomi из одноименного модуля ядра, другие модемы сразу же распознаются как оборудование для последовательной передачи данных. После этого модемом можно будет управлять через специальный файл-устройство (например, /dev/ttyUSB*n*, /dev/ttyACM*n* или /dev/nozomi*n*). С точки зрения Linux, современный UMTS-модем ничем не отличается от аналогового модема, выпущенного 20 лет назад! Как и ранее, для работы требуются команды AT. Для управления некоторыми функциями, характерными для мобильной связи, разработаны новые AT-команды (например, AT+CPIN=*nnnn* для передачи PIN-кода).

Поэтому в принципе возможно использовать UMTS-модемы с программами gnome-ppp или KPPP, которые разрабатывались для управления аналоговыми модемами. Чтобы этот метод функционировал, нужно предварительно отключить запрос PIN-кода. Методом следует пользоваться лишь в крайнем случае.

Дополнительная функция: использование в качестве носителя для хранения данных. У большинства распространенных UMTS-модемов есть дополнительная функция хранения информации. При подключении к компьютеру они сначала отображаются как USB-носитель данных. Таким образом, с них можно с легкостью установить драйвер для работы с Windows или OS X, поставляемый вместе с модемом. Чтобы активизировать саму функцию модема (то есть превратить съемный носитель в модем), нужно выполнить специальный код.

В Linux такая неоднозначность может приводить к проблемам. Если модель устройства известна, ядро Linux посылает таким модемам специальную байтовую последовательность, позволяющую активизировать функцию модема. За этот автоматический процесс отвечает команда `usb_modeswitch`, содержащаяся в одноименном пакете. Команда выполняется после получения верной байтовой последовательности, как только система udev распознает UMTS-модем. Соответствующее правило udev в Ubuntu располагается в файле `/lib/udev/usb_modeswitch`. Заархивированная база данных всех известных модемов и относящихся к ним байтовых последовательностей для переключения находится в файле `/usr/share/usb_modeswitch/configPack.tar.gz`.

Если у вас совсем новый модем и такой автоматический механизм не сработает, то вставьте модем в компьютер и вручную выполните `eject /dev/xxx`, указав имя устройства последнего USB-носителя. Команда `df` выводит список всех носителей данных, из которых потребуется выбрать интересующий вас.

Еще можно попытаться создать собственное правило `usb_modeswitch`. Как правило, байтовая последовательность для переключения в режим модема идентична на устройствах одного и того же производителя. Поэтому зачастую бывает достаточно узнать ID USB, соответствующий модему, с помощью `lsusb` и внести запись в простой файл правил udev.

```
root# lsusb
...
ID 1234:5678 <hersteller abc> HSDPA/HSUPA Modem
```

Теперь нужно создать файл `/lib/udev/rules.d/61-my-usb-modeswitch.conf` по следующему образцу. Всю следующую команду необходимо записать в одну строку. Кроме того, коды `1234` и `5678` нужно заменить на USB-ID вашего модема.

```
ATTRS{idVendor}=="1234", ATTRS{idProduct}=="5678",
   RUN+="modem-modeswitch -v 0x%s{idVendor} -p 0x%s{idProduct} -t
   option-zerocd"
```

Параметры соединения. Обычно в конфигурационных программах UMTS предусмотрены поля для ввода телефонного номера, последовательности символов APN, логина, пароля и PIN-кода. У большинства провайдеров роль телефонного номера играет последовательность символов `*99#`. Последовательность символов APN *(имя точки доступа)* зависит от провайдера и означает точку подключения к мобильной сети. У многих провайдеров поля для ввода логина и пароля можно оставлять пустыми, так как они могут не интерпретироваться.

Проблемы с PIN- и PUK-кодом. Если Network Manager или другие программы постоянно требуют от вас ввода PIN- и PUK-кода, возможны проблемы. Однако

не увлекайтесь экспериментами! Не исключено, что SIM-карта заблокируется после ввода нескольких якобы неправильных PIN-кодов.

PIN-код SIM-карты вполне можно отключить. Проще всего это делается с помощью программы UMTSmon.

Если у вас нет этой программы, достаньте SIM-карту из USB-модема, вставьте в открытый сотовый телефон и с помощью телефона отключите запрос PIN-кода (например, Настройки ▸ Настройки безопасности ▸ Запрашивать PIN-код ▸ Вкл./Выкл.). Затем снова вставьте SIM-карту в USB-модем. Теперь соединение с Интернетом возможно и без PIN-кода. Но не потеряйте ваш модем, так как теперь кто угодно сможет пользоваться им, не вводя PIN-код!

Более подробная информация и советы по работе с UMTS в Linux даны на сайте http://umtsmon.sourceforge.net/docs/OLS.umts.paper.pdf.

18.9. Основы ADSL

Сравнение модема и роутера. При доступе в Интернет с помощью ADSL существует два основных варианта соединения компьютера с Сетью: через ADSL-модем и ADSL-роутер.

- **ADSL-модем.** По сути, он является оконечным устройством сети ADSL (кратко — ATN). К одному ADSL-модему можно подключить только один компьютер. Обмен информацией между вашим компьютером и Интернетом осуществляется через USB-разъем или сетевой кабель. В зависимости от модема и конкретного провайдера при обмене информацией могут применяться протоколы PPPoE, PPPoA или PPTP.

  Из-за обилия вариантов конфигурация может оказаться сложной. Если вам повезет, то настройку удастся выполнить с помощью Network Manager или инструментов, характерных для конкретных дистрибутивов (`pppoe-config` в Debian и Ubuntu), `system-config-network` в Fedora и Red Hat, YaST в SUSE). Если эти инструменты не помогут, то посмотрите подразделы о конфигурации ADSL-PPPoE и ADSL-PPTP, где рассказано, как настраивать соединение вручную.

- **ADSL-роутер.** Совмещает функции ADSL-модема и функции роутера или шлюза. К роутеру с помощью сетевых кабелей можно подключить несколько компьютеров. Компьютеры получают все конфигурационные файлы от ADSL-роутера через DHCP, поэтому отдельно настраивать каждый клиент не требуется. В некоторых ADSL-роутерах также имеется модуль WLAN. Если у вас есть выбор, обязательно работайте с ADSL-роутером!

Существуют также модемы USB-SpeedTouch, которые используются с протоколом PPPoA. Но я не рекомендую работать с такими модемами в Linux. Установка очень сложна, а сам метод чреват многочисленными сбоями (проверено на практике!).

Лучше купить PPPoA-совместимый ADSL-роутер с обычным телефонным входом, а USB-модем не использовать.

Конфигурация ADSL-роутера

Многие провайдеры бесплатно предоставляют роутер при подключении. Если же вам его не предоставили, то настоятельно рекомендую приобрести ADSL-роутер. При этом необходимо обращать внимание на три момента:

- роутер должен поддерживать протокол, используемый вашим провайдером (например, PPPoE);
- роутер должен быть совместим с применяемой ADSL-технологией (например, ADSL2+);
- роутер должен иметь подходящий телефонный разъем (A или B).

A и B — это дополнения к стандарту G.992.1, описывающие параллельное использование каналов передачи (обычного телефонного канала и ADSL). Например, в Германии все подключения ADSL выполняются в соответствии со стандартом B. Данная спецификация позволяет параллельно использовать телефонное соединение с ISDN и ADSL.

В большинстве других государств ADSL-соединение, напротив, обычно устанавливается в соответствии с приложением A. Данная спецификация позволяет параллельно применять аналоговое телефонное соединение и ADSL. Исключение представляют только ISDN-разъемы; если требуется совместно использовать ISDN и ADSL, то во всем мире в таких случаях применяется приложение B.

Перед началом работы ADSL-роутер нужно сконфигурировать. Как правило, вы подключаете компьютер к роутеру, открываете браузер и вводите IP-адрес роутера, то есть, например, `http://192.168.0.101` (адрес указывается в инструкции к роутеру). После этого открывается удобный веб-интерфейс для настройки ADSL. Для создания конфигурации PPPoE здесь нужно указать логин и пароль вашего ADSL-соединения.

Если же ваш провайдер использует PPPoA, то необходимо настроить параметры VPI (Virtual Path Identifier, идентификатор виртуального маршрута) и VCI (Virtual Channel Identifier, идентификатор виртуального канала). Правильные значения зависят от инфраструктуры ATM провайдера и специфичны для провайдера и страны. В следующем списке приведены некоторые распространенные значения:

- Бельгия — VPI=8; VCI=35;
- Дания — VPI=0; VCI=35;
- Италия — VPI=8; VCI=35;
- Франция — VPI=8; VCI=35;
- Великобритания — VPI=0; VCI=38;
- Нидерланды — VPI=8; VCI=48;
- Австрия — VPI=8; VCI=48;
- Испания — различные комбинации, например 1/32,1/33,8/32 и 8/35.

При необходимости узнайте у провайдера, какие значения следует использовать, или выполните в Интернете поиск по ключевым словам `список VPI VCI`.

Конфигурация ADSL-PPPoE

Конфигурация сетевой карты. Для соединения между ADSL-модемом и вашим компьютером используется Ethernet-кабель. При этом в отличие от большинства случаев не применяется протокол TCP/IP. При конфигурации сетевой карты не учитываются ни IP-адрес, ни маска сетевого интерфейса, поэтому настройка сети не требуется. В таком случае сетевой интерфейс нельзя конфигурировать как шлюз!

Конфигурация pppd. PPPoE в Linux обрабатывается с помощью модуля ядра. Чтобы `pppd` могла использовать сетевой интерфейс для PPPoE, используйте плагин `rp-pppoe.so`. В большинстве распространенных дистрибутивов этот файл устанавливается вместе с `pppd`.

Для конфигурации `pppd` вам, как обычно, потребуется конфигурационный файл в `/etc/ppp/peers/`. Для работы следующего кода необходимо, чтобы каталог `/etc/options` был пуст:

```
#/etc/ppp/peers/adsl
#PPPoE-специфичные параметры
pluginrp-pppoe.so
mru1492
mtu1492

# к этому интерфейсу подключается ADSL-модем
eth0

# Обычные параметры
lock
noauth
noipdefault
defaultroute
usepeerdns

# Логин для /etc/ppp/pap-secrets или chap-secrets
name"hofer"

# При разрыве соединения ждать 4 секунды,
# затем попытаться установить новое соединение
persist
holdoff4
maxfail25

#для Red Hat/Fedora
ipparam"adsl"
```

Обычные параметры действуют так же, как и при любых других соединениях с Интернетом, использующих PPP. Пароль для настройки `name` должен находиться в файле `chap-secrets` или `pap-secrets`.

С помощью `persist` соединение может быть автоматически восстановлено после обрыва. Параметр `holdoff` настраивает время ожидания между обрывом соединения

и попыткой восстановить его, а maxfail указывает, после какого максимального количества неудачных попыток соединения pppd должна прекратить эти попытки. Эти параметры полезны в тех случаях, когда требуется бесперебойное соединение с Интернетом.

Если бесперебойное соединение не требуется, добавьте в конфигурационный файл idle *n*. В таком случае, если в течение *n* секунд вы не будете работать с системой, соединение с Интернетом автоматически завершится.

Параметр ipparam используется в Red Hat и Fedora для того, чтобы задействовать автоматическую конфигурацию DNS. В этих дистрибутивах дополнительно требуется следующий файл:

```
#/etc/sysconfig/network-scripts/ifcfg-adsl
PEERDNS=yes
```

Автоматическое тестирование соединения. При определенных обстоятельствах pppd может «не заметить», что у провайдера возникли проблемы с соединением и он больше не отвечает. Следующие дополнения, вносимые в конфигурационный файл, заставляют программу каждые 60 секунд отправлять провайдеру запрос, на который ожидается отклик. Если на два запроса, посланных друг за другом, не удается получить отклик, pppd завершает соединение. Если в конфигурационном файле содержится команда persist, то сразу же после этого делается попытка установить новое соединение.

```
#Дополнение в /etc/ppp/peers/adsl
lcp-echo-interval60
lcp-echo-failure2
```

MTU и MRU. В Интернете данные передаются не байт за байтом, а в виде пакетов. В сети Ethernet такие пакеты по умолчанию имеют размер 1500 байт. Если на пути между адресатом и отправителем данных есть оборудование или программы, для которых эти пакеты слишком велики, они автоматически дробятся на пакеты меньшего размера, а затем снова собираются.

Однако такой метод не слишком эффективен, и поэтому в некоторых операционных системах (в том числе в Linux) предпринимаются попытки определить максимальный размер пригодных для отправки пакетов путем рассылки тестовых ICMP-пакетов. Правда, следует отметить, что такие пакеты «проглатываются» некоторыми брандмауэрами, из-за чего точно определить максимальный размер пакета не удается, а при некоторой доле невезения передача данных вообще прекращается.

Обычно такой проблемы не возникает, так как наименьший общий знаменатель, то есть 1500 байт, превышается редко. Но как раз при использовании PPPoE потери происходят, так как несколько байт из 1500-байтового пакета тратится на протоколирование информации. Поэтому показатель максимальной длины получаемых пакетов (MRU) и максимальной длины передаваемых пакетов (MTU) следует снизить до 1492 байт.

Запуск и остановка pppd. Запуск pppd производится, как и при использовании аналогового модема, с помощью следующей команды:

```
root# pppd call adsl
```

Если вы хотите снова завершить соединение, остановите работу pppd командой killall:

```
root# killall pppd
```

В дистрибутивах, построенных на основе Debian, вместо этой команды применяется pon adsl или poff adsl.

Автоматический запуск ADSL с помощью сценария Init-V. Поскольку ADSL часто используется без ограничений по времени, соединение с Интернетом обычно устанавливается при запуске компьютера и сохраняется вплоть до его выключения.

Лучше всего проделывать эти операции с помощью сценария Init-V, который выполняется при запуске компьютера.

Та часть сценария, которая строится в зависимости от конкретного дистрибутива, может выглядеть, например, так:

```
#/etc/init.d/adsl

#... Команды, специфичные для определенного дистрибутива...

case "$1" in
  start)
    echo "Starting adsl"
    pppd call adsl
    ;;
  stop)
    echo "Shutting down adsl"
    [-f/var/run/ppp-adsl.pid] && \
       kill $(head -1 /var/run/ppp-adsl.pid)
  *)
    echo "Usage: $0 {start|stop}"
    exit 1
    ;;
esac
```

Сценарий отменяет действие команды killall pppd, так как на одном компьютере может действовать несколько процессов pppd. Команда killall остановила бы все такие процессы. Указанный сценарий, напротив, останавливает только тот процесс pppd, чей номер он считывает из файла /var/run/ppp-adsl.pid. При этом система сначала проверяет, существует ли указанный файл. Если это не так, то команда head -1 находит первую строку, содержащую номер процесса, и передает ее kill.

Для того чтобы pppd действительно сохранила свой номер процесса в /var/run/ppp-adsl.pid, в конфигурационный файл pppd необходимо вставить следующую строку:

```
#Дополнение в /etc/ppp/peers/adsl
linkname "adsl"
```

После того как вы протестируете сценарий с помощью /etc/init.d/adsl start или stop, потребуется настроить его автоматический запуск.

MSS Clamping. К сожалению, параметры MTU и MRU для pppd действуют только на локальном компьютере. Если этот компьютер одновременно является интернет-шлюзом для других компьютеров, то и на каждом из клиентских компьютеров, подключенных к шлюзу, настройки MTU также потребуется изменить.

Чтобы облегчить связанную с этим работу по конфигурированию системы, существует хорошее решение, называемое *MSS Clamping — сжатие максимального сегмента данных*. При использовании этого решения параметр MSS (максимальный размер сегмента данных) пакетов с данными, передаваемых по TCP, корректируется в соответствии со значением MTU на локальном компьютере. Чтобы понять, что такое максимальный размер сегмента данных, нужно быть экспертом по TCP.

За реализацию MSS Clamping отвечает система ядра iptables; для этого необходимо активизировать указанное ниже правило брандмауэра. При этом следует заменить eth0 названием интерфейса, к которому подключен ADSL-модем:

```
root# iptables -o eth0 --insert FORWARD 1-p tcp --tcp-flags SYN, RST SYN \
               -m tcpmss --mss 1400:1536 -j TCPMSS --clamp-mss-to-pmtu
```

Как правило, эта команда входит в состав сценария брандмауэра, оборудованного на компьютере-шлюзе. Но вы можете интегрировать это правило и в описанный выше файл сценария Init-V. Правда, в таком случае необходимо убедиться, что при многократном запуске системы ADSL правило выполняется лишь один раз.

Конфигурация ADSL-PPTP

Туннельный протокол типа «точка — точка» со стороны компьютера-клиента поддерживается программой pptp. В большинстве распространенных дистрибутивов эта программа предоставляется в отдельном одноименном пакете, но данный пакет не всегда устанавливается автоматически. Базовая информация о PPTP, а также о графической программе pptpconfig, которая предназначена для конфигурации PPTP, но в большинстве дистрибутивов пока отсутствует, дается на следующем сайте: http://pptpclient.sourceforge.net.

PPTP используется не только для установления ADSL-соединений, но и для создания виртуальных частных сетей. Конфигурация VPN на клиентском компьютере очень напоминает конфигурацию ADSL. Совершенно иначе выглядит конфигурация VPN-сервера, где вместо pptp применяется программа pptpd.

Конфигурация сетевой карты. Поскольку соединение между ADSL-модемом и компьютером осуществляется по Ethernet, сначала нужно настроить именно это соединение. Вам потребуется выбрать IP-адрес и маску для сетевого интерфейса так, чтобы модем был доступен. Если модем имеет IP-адрес 10.0.0.138, как у некоторых моделей Alcatel, то для сетевого интерфейса необходимо задать номер 10.0.0.*n*, причем *n* не должно быть равно ни 0, ни 255, ни 138. В качестве маски используется 255.255.255.0. Настраивать адрес шлюза нельзя (общая ин-

формация о конфигурации сети дается в разделе 16.3). Чтобы проверить, есть ли соединение между вашим компьютером и ANT, выполните обычную команду ping:

```
root# ping 10.0.0.138
PING 10.0.0.138 (10.0.0.138): 56 data bytes
64 bytes from 10.0.0.138: icmp_seq=0 ttl=15 time=4.674 ms
64 bytes from 10.0.0.138: icmp_seq=1 ttl=15 time=3.737 ms
...
```

Конфигурация pppd. Для конфигурации pppd вам, как обычно, требуется специальный конфигурационный файл в /etc/ppp/peers/. Для работы следующего кода необходимо, чтобы каталог /etc/options был пуст:

```
# /etc/ppp/peers/adsl
# pptp-специфичные
pty "/usr/sbin/pptp 10.0.0.138 --nolaunchpppd"

# Опционально: затребовать определенный метод шифрования
# require-mppe-128

# Без сжатия
nobsdcomp
nodeflate

# Логин для /etc/ppp/pap-secrets или chap-secrets
name "hofer"

# Обычные параметры
lock
noauth
noipdefault
defaultroute
usepeerdns

# При обрыве соединения ожидать 4 секунды,
# затем установить новое соединение
persist
holdoff 4
maxfail 25

# Для Red Hat/Fedora
ipparam "adsl"
```

Параметр pty отвечает за вызов pptp, а -nolaunchpppd не позволяет pptp запустить программу pppd. В показанной здесь конфигурации это не обязательно, так как pppd уже работает и, в свою очередь, вызывает pptp.

Пароль для настройки name должен, как обычно, находиться в chap- secrets или pap-secrets. Параметр ipparam необходим для того, чтобы в Red Hat и Fedora работала автоматическая конфигурация DNS. В этих дистрибутивах дополнительно требуется следующий файл:

```
# /etc/sysconfig/network-scripts/ifcfg-adsl
PEERDNS=yes
```

Запуск pppd. При запуске или остановке работы pppd нас уже ничто не удивит:

```
root# pppd call adsl
root# killall pppd
```

Сценарий для автоматического запуска ADSL был описан выше. Его можно в таком же виде использовать и с PPTP. Единственное необходимое условие заключается в том, что сначала требуется настроить сетевое соединение с модемом.

19 Интернет-шлюз

Далее нам предстоит изучить несколько тем, посвященных конфигурации сервера. Было бы самонадеянно попытаться охватить все вопросы конфигурации сервера в Linux. Почти каждый из разделов этой главы можно расширить до целой книги.

Я писал главы о конфигурации сервера для того, чтобы сообщить вам базовую информацию по данной теме, исключительно важной для опытных пользователей Linux. Эти главы адресованы, в первую очередь, специалистам, которые занимаются администрированием относительно небольших локальных сетей, причем в таких сетях вполне могут находиться и компьютеры-клиенты с Windows.

ADSL-роутер… В этой главе речь пойдет о том, как создать интернет-шлюз для локальной сети. В частных сетях эту задачу часто выполняет ADSL-роутер. Такой метод имеет очевидные достоинства: конфигурация проста, устройство работает бесшумно и потребляет относительно немного энергии. В общем, если вас устраивает ADSL-роутер, то продолжайте с ним работать, а эту главу можете пропустить! Так вы сэкономите себе массу времени и сил.

…и собственный шлюз. И все же есть случаи, когда конфигурационных возможностей ADSL-модема явно недостаточно: в некоторых моделях брандмауэров предусмотрено слишком мало параметров, разрешение имен не соответствует требованиям Kerberos, интегрированный DHCP-сервер оказывается непригоден для создания защищенной конфигурации WLAN/VPN и т. д. В результате приходится обратиться к этой главе, то есть настроить отдельный компьютер, который будет выполнять следующие функции ADSL-роутера: маскарадинг, работа DHCP-сервера и сервера имен. Любой интернет-шлюз обязательно должен быть защищен брандмауэром. О том, как это сделать, рассказано в главе 26.

19.1. Введение

В этой главе будет описана установка следующих компонентов (служб).

- **Маскарадинг/NAT.** С помощью маскарадинга можно соединить все локальные клиенты, подключенные к Интернету. Итак, есть компьютер с Linux, устанавливающий соединение с Интернетом посредством ADSL или ISDN. Все остальные компьютеры сети соединены с этим компьютером и также могут пользоваться Интернетом. Отпадает необходимость оборудовать каждый компьютер отдельным модемом!

- **DHCP и DNS.** DHCP обеспечивает простое централизованное администрирование IP-адресов и других сетевых параметров всех клиентов. Локальный сервер имен гарантирует, что клиенты, в свою очередь, знают свои имена (разрешение локальных имен в IP-адресах). Кроме того, программа играет роль кэша IP-адресов и последующие соединения с Интернетом становятся немного быстрее.

  Здесь будут рассмотрены два варианта реализации функций DHCP и DNS. Проще всего воспользоваться командой `dnsmasq`, которая сочетает в себе обе функции и проста в конфигурировании. Если же вы предъявляете к сети повышенные требования или строите большую и сложную локальную сеть, рекомендуется распределить между двумя программами `dhcpd` и `bind`. Таким образом повышаются возможности конфигурации, и вы можете выполнить ее с большей легкостью.

- **Интеграция с WLAN.** Соединить обычную сеть и WLAN совсем не сложно. Проблема заключается в том, чтобы обеспечить безопасную реализацию функций WLAN. Поэтому в последнем разделе этой главы будут рассмотрены некоторые варианты конфигурации.

Оборудование. Если интернет-шлюз не должен выполнять никаких функций, кроме задач, собственно, «шлюза», то его можно создать на медленном ПК с минимумом оборудования и этого будет вполне достаточно. Однако в компьютере должно быть два сетевых интерфейса: один — для подсоединения компьютера к ADSL-роутеру или к ADSL-модему, а другой — для подключения к локальной сети.

Когда вы займетесь поиском тихого, недорогого и энергосберегающего компьютера, вы быстро убедитесь, что самое большое препятствие — второй сетевой интерфейс: мини-ПК, например Eee Box или MSI Wind PC, обычно имеют только один сетевой интерфейс, и к ним нельзя подключать сменные платы (PCI, PCIe). Один из возможных выходов — использовать USB-Ethernet-адаптер, но пока сделать это в Linux сложно. Мне понравилось применять устройство DUB-E100 фирмы D-Link, которое начинает работать «с ходу», без дополнительной настройки. Список подходящих устройств приводится по адресу http://www.linux-usb.org/usbnet/.

Интересной альтернативой является WLAN-роутер WRT54GL фирмы Linksys или совместимые с ним устройства. Характерной чертой этих устройств является то, что их «прошивка» существует в версии для Linux, которую можно и нужно использовать. в Интернете представлен целый спектр подходящих дистрибутивов (Tomato, DD-WRT, OpenWRT и т. д.). Конечно же, чтобы начать работу с этим оборудованием, придется немного потрудиться. Результатом вашего труда будет дешевый, полностью бесшумный прибор, потребляющий значительно меньше электричества, чем мини-ПК, имеющиеся на рынке.

Установка пакетов. В целях безопасности пакеты серверных служб, описанные в этой главе, по умолчанию *не* устанавливаются. Иногда вам могут встретиться пакеты с похожими названиями для соответствующих клиентских служб; но в данном случае они не подойдут. После установки серверных пакетов программы также следует специально активировать.

Помощь в конфигурации. Эта глава адресована опытным пользователям Linux. Здесь описаны лишь настройки различных конфигурационных файлов, но не рассмотрены пользовательские интерфейсы, которые могут помочь вам при конфигурации и могут находиться в свободном доступе. В Fedora и Red Hat это различные команды `system-config-xxx`, в SUSE — модули **Службы сети** в YaST.

Здесь будет рассмотрен метод работы с изменением конфигурационных файлов вручную — он может показаться старомодным, но на практике вполне себя оправдывает: лишь в том случае, если вы выполняете конфигурацию вручную, вы знаете, где находятся конфигурационные файлы. И только используя этот способ, вы можете достаточно быстро сориентироваться в готовой конфигурации другого сервера (например, при новой установке системы или смене дистрибутива).

Безопасность. Если вы хотите использовать Linux на сетевом сервере, то просто *обязаны* хорошо изучить тему безопасности — несерьезное отношение к этой теме является грубой халатностью. Советы, касающиеся основ обеспечения безопасности сервера, даются в главе 18.

IPv6. Все программы, описанные в этой главе, совместимы с IPv6. Однако я тестировал сетевые функции лишь с IPv4, поэтому в дальнейшем особенности IPv6 рассматриваться не будут.

Примерная топология сети. Чтобы вы могли лучше ориентироваться в этой главе, на рис. 19.1 показана примерная топология сети. В этой локальной сети используется адресное пространство 192.168.0.* и доменное имя **sol**. Компьютер-шлюз с хост-именем **mars** имеет постоянный адрес 192.168.0.1. Соединение с Интернетом осуществляется через ADSL-роутер.

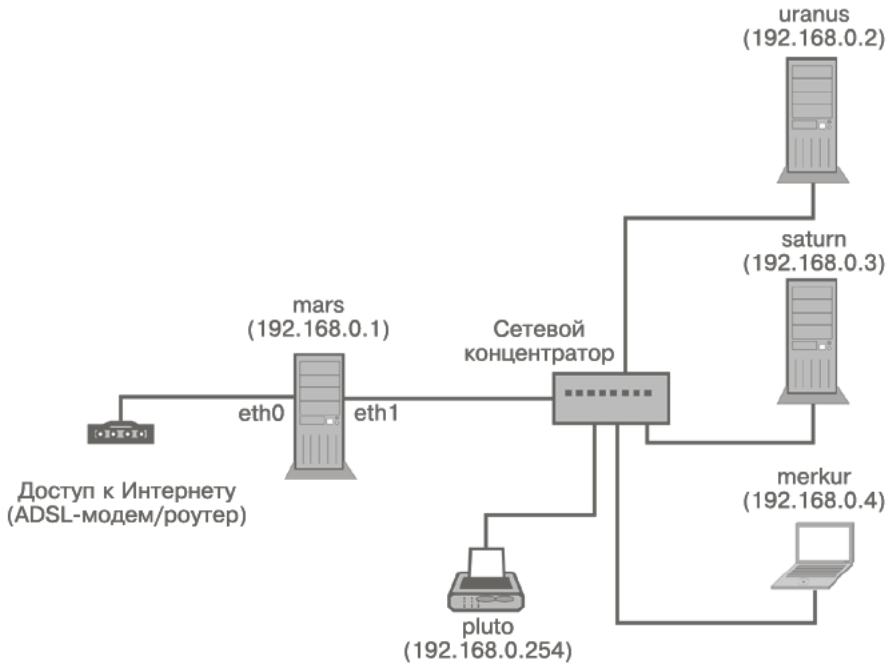


Рис. 19.1. Примерная топология сети

Клиенты локальной сети соединены с компьютером mars через сетевой концентратор. Клиентам динамически присваиваются IP-адреса (192.168.0.2–192.168.0.253). Единственное исключение представляет собой сетевой принтер pluto, которому присвоен статический IP-адрес 192.168.0.254.

Для обеспечения безопасности рекомендуется оборудовать выход в Интернет с компьютера mars брандмауэром. Но по причинам, связанным с экономией энергии и денег, в небольших и частных сетях все же целесообразно интегрировать все серверные функции на одном компьютере, как это описано ниже.

19.2. Сетевая конфигурация

Сетевая конфигурация интернет-шлюза обычно осуществляется вручную, как это было описано в разделе 18.5. Но у интернет-шлюза есть характерная особенность, заключающаяся в том, что он, на самом деле, состоит из двух интерфейсов. Один устанавливает соединение с Интернетом, другой — с локальной сетью.

Debian, Ubuntu

В Debian и Ubuntu файл `/etc/network/interfaces`, отвечающий за конфигурацию, должен выглядеть приблизительно, как в следующем примере:

```
# /etc/network/interfaces
auto lo
iface lo inet loopback

# Динамическое соединение с DHCP-сервером,
# сообщающим основные показатели для доступа в Интернет
auto eth0
iface eth0 inet dhcp

# Статическая конфигурация для соединения с LAN
auto eth1
iface eth1 inet static
  address 192.168.0.1
  netmask 255.255.255.0
```

Если соединение с Интернетом является статическим, то файл `interfaces` нужно доработать следующим образом. В примере предполагается, что соединение осуществляется через ADSL-роутер, имеющий IP-адрес 10.0.0.138 (как устройство SpeedTouch, которым пользуюсь я). Этот адрес одновременно является адресом шлюза, через который осуществляется выход в Интернет (ключевое слово `gateway`).

```
# /etc/network/interfaces
...
# Статическое соединение с ADSL-роутером,
# имеющим IP-адрес 10.0.0.138
auto eth0
iface eth0 inet static
  address 10.0.0.1
```

```
netmask 255.255.255.0
gateway 10.0.0.138
```

Fedora, Red Hat

В Fedora и Red Hat вам понадобится по одному конфигурационному файлу `ifcfg-xxx` в каталоге `/etc/sysconfig/network-scripts` на каждый интерфейс, где `xxx` — имя интерфейса. Оба конфигурационных файла должны строиться по следующему образцу:

```
# Файл /etc/sysconfig/network-scripts/ifcfg-eth0
DEVICE=eth0
HWADDR=xx:xx:xx:xx:xx:xx (Собственный MAC-адрес)
NM_CONTROLLED=no
ONBOOT=yes
BOOTPROTO=dhcp
TYPE=Ethernet
USERCTL=no
PEERDNS=yes
IPV6INIT=no
```

19.3. Маскарадинг (NAT)

Исходной точкой для маскарадинга служит компьютер, который уже подключен к Интернету (в этой главе — компьютер mars). Теперь мы должны дать доступ в Интернет и всем другим компьютерам, находящимся в локальной сети.

Компьютеры в локальной сети используют частные IP-адреса: эти адреса расположены в специально зарезервированных адресных пространствах (например, 10.*.*.* или 192.168.*.*). Теперь адреса являются уникальными только в рамках сети LAN, но не в Интернете. Поэтому интернет-шлюз не может просто переадресовывать запросы страниц в LAN.

Принцип *маскарадинга* заключается в том, что компьютер-шлюз принимает пакеты, направленные клиентами в Интернет, и изменяет их обратные адреса так, как будто эти пакеты были посланы самим компьютером-шлюзом. Такое изменение адреса также называется *трансляцией сетевых адресов* (Network Address Translation, NAT).

Теперь пакет с данными может быть направлен по определенному адресу в Интернете. Как правило, через некоторое время из Интернета приходит ответ — например, отображается сайт. Шлюз должен переадресовать ответ тому клиенту, который отправлял соответствующий запрос. Для этого шлюз должен «отгадать» правильный адрес получателя, ведь пакет (в соответствии с изменением адреса) был отправлен самим шлюзом, поэтому и ответ приходит именно на шлюз.

Чтобы обеспечить правильное присвоение адресов пакетам, полученным в ответ, шлюз изменяет не только обратный адрес, но и порт отправителя. Для каждого IP-адреса, находящегося в сети, применяется определенный номер порта. Внутри системы Linux за маскарадинг отвечает система `iptables`. Это интегрированная в ядро система, предназначенная для обработки IP-пакетов.

Если компьютер-шлюз подключается к Интернету через ADSL-роутер (а не через модем), то это устройство еще раз осуществляет маскарадинг, то есть манипулирует адресами. К счастью, никаких проблем из-за этого не возникает.

Функции маскарадинга и брандмауэра тесно связаны друг с другом. В этом разделе предполагается, что компьютер-шлюз *не выполняет* функций брандмауэра. Если он их все же выполняет, то вам может потребоваться создать брандмауэрные функции с помощью специального инструмента (на ваш выбор) либо отключить некоторые функции брандмауэра, прежде чем активировать маскарадинг. Базовая информация о том, что такое брандмауэр и как он работает, дается в главе 26.

Понятие о клиенте и сервере. Далее по тексту компьютер, имеющий выход в Интернет, будет называться *сервером*, а все остальные компьютеры — *клиентами*, независимо от того, какие именно функции выполняют эти компьютеры. При осуществлении маскарадинга сервер зачастую будет называться *интернет-шлюзом* (что более правильно), или *интернет-роутером*. Это различие между сервером и клиентом справедливо для всей книги, но только в рамках данной отдельной функции! Компьютер, являющийся клиентом относительно своего выхода в Интернет, вполне может быть сервером при выполнении другой функции (например, NFS). На практике очень часто несколько серверных функций реализуется на одном компьютере. Но такая концентрация не является обязательной, а в больших сетях — ради повышения эффективности работы сети — такой концентрации даже следует избегать.

Включение и выключение маскарадинга

Red Hat. В некоторых дистрибутивах функцию маскарадинга можно активизировать в программе конфигурации брандмауэра. В RHEL запустите для этого программу `system-config-firewall`. На вкладке Доверенный интерфейс выберите интерфейс для LAN. Затем убедитесь, что брандмауэр не блокирует трафик, идущий в локальную сеть. Кроме того, укажите на вкладке Маскарадинг интерфейс, через который будет осуществляться подключение к Интернету.

SUSE. В SUSE необходимо соотнести интерфейс LAN и внутреннюю сеть — для этого используйте функцию Безопасность ▸ Брандмауэр в модуле YaST. Дополнительно установите на вкладке Маскарадинг одноименный флажок.

Debian, Ubuntu. В Debian и Ubuntu пока нет специальных конфигурационных инструментов для создания брандмауэра и обеспечения маскарадинга. Но вы можете, например, установить программу FireStarter и активировать с ее помощью и брандмауэр, и маскарадинг.

Активизация вручную. Если вы хотите управлять маскарадингом, не используя инструментария, связанного с брандмауэром, выполните две короткие команды:

```
root# echo 1 > /proc/sys/net/ipv4/ip_forward
root# iptables -A POSTROUTING -t nat -o eth0 -j MASQUERADE
```

Команда `sysctl` активизирует в ядре функцию IP-переадресации, которая по умолчанию отключена из соображений безопасности. Если команда `sysctl` в вашем дистрибутиве отсутствует, можете воспользоваться следующей командой:

```
root# echo 1 > /proc/sys/net/ipv4/ip_forward
```

Команда iptables определяет правило, в соответствии с которым IP-пакеты, покидающие локальную сеть, должны перенаправляться на интерфейс eth0 и претерпевать при этом преобразования в соответствии с правилами NAT. В зависимости от конфигурации доступа к Интернету вам может понадобиться указать и другой интерфейс, например ppp0. (Под eth0 в этой главе понимается интерфейс, через который компьютер подключается к Интернету.)

Деактивация вручную. Чтобы отключить функции маскарадинга, выполните следующие команды:

```
root# iptables -t nat -D POSTROUTING -o eth0 -j MASQUERADE
root# sysctl -w net.ipv4.ip_forward=0
```

Вместо sysctl может сработать также следующая команда echo:

```
root# echo 0 > /proc/sys/net/ipv4/ip_forward
```

Автоматическая активизация маскарадинга. Разумеется, вам не придется активизировать маскарадинг при каждом запуске компьютера. Обычно маскарадинг запускается вместе с брандмауэром. Если вы настраивали свой брандмауэр вручную, активизируйте его вместе с функцией маскарадинга с помощью сценария, выполняемого в процессе Init-V (подробнее об этом читайте в главе 16). В некоторых дистрибутивах может применяться сценарий Init-V, представленный в пункте «Как построены файлы сценариев Init-V» раздела 16.4. Не забудьте, соответственно, активизировать сценарий с помощью chkconfig (Red Hat), insserv (Debian, SUSE) или update-rc.d (Ubuntu)!

Если вы подключаетесь к Интернету по PPP (например, через ADSL-модем), то можете интегрировать команды маскарадинга в сценарии ip-up и ip-down в /etc/ppp. Таким образом, маскарадинг активизируется одновременно с подключением к Интернету и автоматически деактивируется, когда соединение разрывается.

Независимо от применяемой системы брандмауэра, в большинстве дистрибутивов можно на постоянной основе активизировать IP-переадресацию. Для этого нужно добавить специальную запись в файл /etc/systcl.conf. Этот файл интерпретируется при каждом запуске компьютера.

```
 # в /etc/sysctl.conf
...
net.ipv4.ip_forward = 1
```

Проблемы

Бесспорно, маскарадинг — это достаточно изящное решение, обеспечивающее общий доступ из локальной сети в Интернет. Но с маскарадингом могут возникать определенные проблемы.

- Существует несколько интернет-протоколов, в которых предусмотрены защитные механизмы, проверяющие распределение IP-адресов. При использовании маскарадинга соотнесенность между IP-адресом и компьютером перестает быть однозначной, и это может мешать работе.
- В некоторых протоколах предусмотрена возможность передачи IP-адресов не только в IP-пакетах, но и в пакетах с данными (в виде текста ASCII или

в зашифрованном виде). Наиболее известный пример такого протокола — FTP. Чтобы FTP работал, несмотря на применение маскарадинга, маскарадинговый сервер должен изменять не только IP-адреса пакетов, но и их содержание.

В Linux для множества интернет-служб предусмотрены специальные маскарадинговые модули (например, `nf_nat_ftp` для FTP). Модули загружаются автоматически. Если они не загружаются, активизируйте их с помощью команды `modprobe`:

```
root# modprobe nf_nat_ftp
root# modprobe nf_conntrack_ftp
```

Если при подключении FTP-клиентов к Интернету по-прежнему возникают проблемы, их можно устранить, переведя работу в пассивный режим. Большинство FTP-клиентов автоматически активизируют этот режим либо предусматривают возможность настроить его вручную.

- Если маскарадинговый сервер соединен с Интернетом по ADSL/PPPoE, могут возникать сложности с максимальной длиной пакетов. Чтобы разрешить эту проблему, можно уменьшить во всех пакетах максимальную длину (MTU) либо задействовать на сервере метод MSS Clamping. Для этого необходимо выполнить следующую команду `iptables`. С ее помощью к слишком большим пакетам применяется MSS Clamping.

```
root# iptables -I FORWARD -p tcp --tcp-flags SYN,RST SYN \
      -j TCPMSS --clamp-mss-to-pmtu
```

Конфигурация клиента

Для того чтобы клиент мог использовать выход в Интернет, предоставляемый Linux-шлюзом, при сетевой конфигурации клиентов необходимо учесть два момента.

- В качестве адреса шлюза нужно указать IP-адрес Linux-шлюза (при топологии сети, как на рис. 19.1, это будет адрес 192.168.0.1).
- Адрес сервера имен должен совпадать с IP-адресом сервера имен для Linux-шлюза. Этот адрес присваивается провайдером интернет-услуг и содержится в файле `/etc/resolv.conf` Linux-шлюза.

19.4. Основы работы с DHCP и сервером имен

DHCP. Разумеется, можно настроить сетевые параметры для каждого компьютера локальной сети отдельно. Но это не только очень утомительно, но и чревато ошибками. Ко всему прочему, вы будете вынуждены выполнить массу дополнительной работы, если решитесь изменить топологию сети.

Гораздо разумнее построить сеть так, чтобы *один из компьютеров* отвечал за присвоение IP-адресов и других параметров всем остальным компьютерам в сети. Для этого используется *протокол динамической настройки хостов* (Dynamic Host

Configuration Protocol, DHCP). Управляющий компьютер становится DHCP-сервером, а все остальные компьютеры — DHCP-клиентами. Есть два основных варианта конфигурации.

- **Динамическая конфигурация** — для клиента настраивается только хост-имя. DHCP-сервер отвечает за все остальные конфигурационные параметры, то есть присваивает клиентам IP-адрес, адрес шлюза, адрес сервера имен и т. д. IP-адреса берутся из специального адресного пула, причем если клиентскому компьютеру потребуется адрес, то DHCP-сервер автоматически присвоит ему ближайший адрес, свободный в настоящий момент.
- **Статическая конфигурация** — при использовании такого варианта конфигурации DHCP-сервер идентифицирует клиентов по ID-номеру сетевой карты. Таким образом, всякий раз компьютер получает один и тот же IP, а в определенных случаях — и хост-имя. Подобный вариант конфигурации несколько сложнее настроить, но при его использовании вы можете применять неизменяющиеся IP-адреса и централизованно управлять хост-именами.

Как правило, первый вариант проще второго. Но вы можете и комбинировать оба варианта — например, чтобы гарантировать, что принтер в любом случае будет иметь один и тот же адрес.

Внутренняя организация DHCP. Принцип работы DHCP примерно таков: если компьютер (то есть DHCP-клиент) перезапускается, он осуществляет широковещательную трансляцию по адресу 255.255.255.255 (таким образом, запрос отправляется на *все* компьютеры, находящиеся в локальной сети). DHCP-сервер реагирует на такой запрос и отсылает в ответ по адресу клиента список доступных IP-адресов.

Куда же сервер посылает ответ, ведь у клиента еще нет IP-адреса? Дело в том, что для адресации достаточно знать MAC-адрес, а он на момент запроса уже известен.

DHCP-сервер выдает IP-адреса на определенный промежуток времени *(время аренды)*. Обычно этот промежуток равен одному дню, но вы можете установить любое значение. До того как закончится время аренды, клиент должен обновить адрес на DHCP-сервере или запросить новый адрес.

Сервер имен. Сервер доменных имен (сервер имен, или DNS) обеспечивает взаимное соответствие между именами компьютеров и IP-адресами. Каждый провайдер интернет-услуг предоставляет своим клиентам DNS, и этот сервер узнает, какой IP-адрес подходит к имени компьютера. Чтобы не обращаться постоянно к DNS, можно настроить в локальной сети собственный сервер имен. В таком случае будет два преимущества.

- **Более высокая скорость.** DNS управляет кэшем недавно использовавшихся интернет-адресов. Иначе говоря, если вы желаете попасть на сайт yahoo.com, на котором уже побывали недавно, вам не нужно повторно обращаться к DNS вашего провайдера, чтобы узнать IP-адрес Yahoo!. Локальный DNS уже запомнил этот адрес. (Кроме того, в большинстве браузеров также задействуется кэш с IP-адресами хост-имен, к которым вы обращались в последнее время. Правда, выигрыш в скорости при использовании такого механизма получается незначительным.)

- **Локальное разрешение имен.** DNS управляет именами и IP-адресами компьютеров, находящихся в локальной сети. Таким образом вы знаете все компьютеры, находящиеся в сети, «поименно» и можете, например, выполнить на компьютере merkur команду `ping saturn`. Тогда компьютер merkur свяжется с локальным сервером имен, который сообщит вам IP-адрес компьютера saturn. Локальное разрешение имен — основная предпосылка для удобного конфигурирования и использования в локальной сети таких служб, как NFS, FTP, SSH и др.

В мире применяется бесчисленное множество DNS, и все они связаны друг с другом. Иными словами, если сам DNS не знает имени, то он может послать соответствующий запрос другому DNS. Все DNS организованы иерархически.

19.5. Программа dnsmasq (DHCP и сервер имен)

В этом разделе мы поговорим о программе `dnsmasq`, в которой интегрированы одновременно сервер имен и DHCP-сервер. Настраивать эту программу довольно несложно. Мощности `dnsmasq` вполне хватает даже для обслуживания больших локальных сетей.

Наиболее популярными альтернативами `dnsmasq` являются программы `bind` и `dhcpd`, реализующие, соответственно, вышеупомянутые серверные функции. Следует отметить, что именно `bind` является сейчас доминирующей программой-сервером имен и, соответственно, центральным элементом всей структуры Интернета. Она применяется, в первую очередь, в крупнейших узлах Интернета, у больших интернет-провайдеров и хостинг-провайдеров. Но эта программа плохо подходит для работы с локальными сетями из-за своей довольно запутанной конфигурации.

Условия

В дальнейшем предполагается, что на вашем компьютере установлена программа `dnsmasq`, а `dhcpd` и `bind` *не* установлены (иначе возникнет конфликт).

Другой важной предпосылкой является правильная конфигурация файла `/etc/hosts` на компьютере-шлюзе. Я на собственном опыте убедился, что этот файл зачастую не соответствует требованиям `dnsmasq`. Не забывайте лишний раз изучить этот файл, прежде чем соберетесь изменять локальную конфигурацию сети, заданную на компьютере-шлюзе!

Как минимум в `/etc/hosts` должны содержаться две приведенные далее строки. Вторая строка имеет определяющее значение для соотнесения имени локального компьютера (например, mars или mars.sol в данном случае) и IP-адреса в локальной сети (здесь 192.168.0.1)! Часто далее присутствуют и другие строки, описывающие конфигурацию IPv6, но мы не будем рассматривать это.

```
# /etc/hosts на компьютере-шлюзе
127.0.0.1    localhost
192.168.0.1  mars       mars.sol
```

СОВЕТ

Если возникнут проблемы, проверьте конфигурацию брандмауэра. DNS-запросы выполняются через TCP- и UDP-порты 53. DHCP использует UDP-порты 67 и 68. Эти порты на сетевом интерфейсе не должны блокироваться брандмауэром.

Файл dnsmasq.conf

Конфигурация `dnsmasq` выполняется в файле `/etc/dnsmasq.conf`. Этот файл, по умолчанию присутствующий в дистрибутиве, также служит техническим документом и содержит около 400 строк комментариев. Можно удалить знаки комментария перед немногочисленными командами, содержащимися в этом файле и активизировать интересующие вас строки. Но гораздо лучше будет создать резервную копию `dnsmasq.conf` и начать работу с нового пустого конфигурационного файла.

```
root# mv /etc/dnsmasq.conf /etc/dnsmasq.conf.orig
root# touch /etc/dnsmasq.conf
```

Для конфигурации `dnsmasq` очень важен не только файл `/etc/dnsmasq.conf`, но также `/etc/hosts` и `/etc/resolv.conf`.

Кроме того, в актуальных версиях Debian и Ubuntu учитываются все конфигурационные файлы, содержащиеся в `/etc/dnsmasq.d`. Необходимо также отметить, что в файле `/etc/default/dnsmasq` имеются некоторые базовые настройки для `dnsmasq` (в частности, `ENABLED=1`, поэтому по умолчанию `dnsmasq` запускается DNS-системой).

Запуск/перезапуск

Как и большинство других программ, представленных в этой и в следующих главах, `dnsmasq` является демоном (системной службой). В некоторых дистрибутивах эта программа запускается сразу же после установки. Если этого не происходит, программу нужно запустить вручную. Все изменения конфигурации вступают в силу только после перезапуска:

```
root# service dnsmasq restart
```

Кроме того, от конкретного дистрибутива зависит, должна ли программа в дальнейшем автоматически запускаться при старте компьютера. В Fedora, Red Hat и SUSE по умолчанию этого не происходит. При работе вам пригодятся следующие команды:

```
root# chkconfig --level 35 dnsmasq on    (RHEL)
root# systemctl enable dnsmasq.service   (OpenSUSE, Fedora)
root# insserv dnsmasq                    (Ранние дистрибутивы SUSE)
```

Минимальная конфигурация

Файл `dnsmasq.conf` достаточно хорошо функционирует уже при показанной ниже минимальной конфигурации. При этом программа играет роль кэша сервера имен

для работы в Интернете и предоставляет клиентам IP-адреса из диапазона между 192.168.0.2 и 192.168.0.250. Клиенты сохраняют свои хост-имена.

```
# /etc/dnsmasq.conf (минимальная конфигурация)
domain-needed
bogus-priv
interface=eth1
dhcp-range=192.168.0.2,192.168.0.250,24h
```

Кратко объясню отдельные ключевые слова: `domain-needed` и `bogus-priv` не позволяют `dnsmasq` передавать хост-имена и IP-адреса локальных компьютеров серверу имен вашего интернет-провайдера (таким образом, управляемый провайдером сервер имен отвечает только за имена и адреса из Интернета, но не из локальных сетей).

Ключевое слово `interface` указывает, что `dnsmasq`, выполняющая функцию DHCP-сервера, должна отвечать только на запросы, приходящие с интерфейса `eth1`, который при рассматриваемой топологии отвечает за LAN. Если `dnsmasq` должна реагировать на запросы от нескольких интерфейсов, эти интерфейсы можно перечислить (`interfaces=eth1,eth2`).

В качестве альтернативы `except-interfaces=...` позволяет указывать интерфейсы, которые `dnsmasq` *не должна* учитывать. Это бывает целесообразно, когда вы конфигурируете интернет-шлюз одновременно и в качестве VPN-сервера (см. главу 27). В таком случае `dnsmasq`, выступающая в качестве сервера имен, будет реагировать и на запросы, которые поступают от интерфейсов `pppn`.

Слово `dhcp-range` указывает, какой диапазон адресов должен использовать DHCP-сервер для ответа на DHCP-запросы. Выданные адреса остаются присвоенными конкретным компьютерам в течение 24 часов, а затем должны быть обновлены клиентом.

Адреса сервера имен и шлюза не требуют дополнительной конфигурации. Программа `dnsmasq` сама интерпретирует `/etc/resolv.conf` и обращается к указанному в этом файле серверу имен. Клиентам DHCP передаются адреса сервера имен и шлюза — отдельно на IP-адрес каждого компьютера в локальной сети.

Применение локального сервера имен

При использовании вышеуказанной минимальной конфигурации `dnsmasq` может разрешать локальные адреса лишь в том случае, если в `/etc/hosts` содержится необходимая информация. Сервер имен не будет знать адресов, которые были присвоены динамически через DHCP. Чтобы `dnsmasq` также могла функционировать в качестве сервера имен для клиентов в сети LAN, добавьте в `dnsmasq.conf` следующие строки и укажите `dnsmasq` заново считать конфигурационный файл. Доменное имя рассматриваемой сети — `sol`.

```
# /etc/dnsmasq.conf  (Применение в качестве сервера имен
# для адресов в локальной сети)
...
local=/sol/
domain=sol
expand-hosts
```

Ключевое слово local указывает, что запросы адресов, поступающие из этого домена, должны обрабатываться непосредственно dnsmasq (а не сервером имен провайдера интернет-услуг).

Слово domain говорит о том, что dnsmasq должна присвоить DHCP-клиентам заданное доменное имя. Это имя должно быть таким же, как и указанное в local.

Наконец, благодаря использованию expand-hosts запросы сервера имен, в которых не указан домен, автоматически дополняются доменным именем, заданным в domain. Итак, если выполнить команду ping uranus, то dnsmasq возвратит адрес uranus.sol.

Чтобы dnsmasq поименно различала своих клиентов, то есть компьютеры, которые запрашивают IP-конфигурацию через DHCP, необходимо, чтобы эти компьютеры сообщили программе свои хост-имена через DHCP. В большинстве дистрибутивов и конфигурационных программ для работы с LAN/WLAN, в том числе в Network Manager, эта настройка действует по умолчанию. Будьте внимательны, применяя на компьютерах-клиентах Fedora, Red Hat или старые версии Debian и Ubuntu. Соответствующие советы о том, как проводить конфигурацию, даются в подразделе «Клиентская конфигурация» раздела 19.5.

Статические адреса и хост-имена

Команду dnsmasq можно сконфигурировать и так, чтобы она настраивала хост-имена клиентов. Статическое соотнесение хост-имени и IP-адреса осуществляется на основании MAC-адреса клиента. Такая адресация удобна, прежде всего, при работе с устройствами, настраивать для которых хост-имена достаточно нелегко — это касается, например, сетевых принтеров. Для конфигурации используется ключевое слово dhcp-host. Следующий листинг — это запись о сетевом принтере pluto.

```
# /etc/dnsmasq.conf     (Статическое присвоение адреса)
...
dhcp-host=00:c0:ee:51:39:9f,pluto,192.168.0.254
```

Самое сложное при таком варианте конфигурации — определить MAC-адрес (адрес для *управления доступом к среде*) клиента. Роль такого адреса играет уникальный ID-номер, который имеется у каждого Ethernet-контроллера. В Linux для отображения MAC-адреса используется команда iconfig. В остальном вам потребуется просто подключить устройство к LAN и выполнить динамическую конфигурацию с помощью dnsmasq. Программа протоколирует все динамические IP-адреса вместе с хост-именами и MAC-адресом в файл /var/lib/misc/dnsmasq.leases. Таким образом, MAC-адрес можно узнать и из этого файла.

По умолчанию адреса, динамически присвоенные с помощью DHCP, действительны в течение 24 часов. Но для сетевых принтеров и другой аппаратуры, работающей в режиме с резервированием, такого промежутка времени часто бывает недостаточно. Если принтер не используется в течение более чем 24 часов, команда dnsmasq считает, что прибор выключен, и как бы забывает о нем. Чтобы этого не происходило, можно добавить к данным о сроке действия адреса дополнительную строку, в которой указывается время. Благодаря infinite адрес остается действителен сколь угодно долго.

```
# /etc/dnsmasq.conf  (Статическое присвоение адреса без ограничения времени)
...
dhcp-host=00:c0:ee:51:39:9f,pluto,192.168.0.254,infinite
```

В man dnsmaq и /etc/dnsmasq.conf.orig описано несколько других вариантов синтаксиса для dhcp-host. Там рассказано, как создать жесткую связь между хост-именем и IP-адресом, полностью заблокировать определенные MAC-адреса и т. д.

DNS для локального компьютера

По умолчанию dnsmasq выполняет функции сервера имен для всех компьютеров в сети, но на самом шлюзе эта программа не работает! Причина заключается в том, что на локальном компьютере используется тот сервер имен, который задан в /etc/resolv.conf. Как правило, этот файл указывает на сервер имен вашего интернет-провайдера или роутера.

Если на компьютере-шлюзе должны работать другие серверные программы (файловый сервер, Kerberos и т. д.), необходимо, чтобы шлюз знал клиентов поименно, то есть также использовал dnsmasq как сервер имен. Чтобы этот механизм работал, обратите внимание на следующие моменты.

- /etc/resolv.conf должен указывать на dnsmasq (то есть на localhost, адрес 127.0.0.1), а не на внешний сервер имен. Внимание: если вы строите соединение с Интернетом по динамическому принципу (LAN + DHCP или модем + PPP), файл resolv.conf будет перезаписан при любой архитектуре соединения. Этого нужно избежать: соединение с ADSL-роутером должно быть статическим. Можете изменить конфигурацию PPP, чтобы в описанном случае resolv.conf оставался без изменений.
- Теперь dnsmasq не может интерпретировать resolv.conf, чтобы узнать адрес внешнего сервера имен, поэтому адрес внешнего сервера имен нужно специально указать в dnsmasq.conf с помощью ключевого слова server.

Продемонстрирую это на примере. Допустим, компьютер-шлюз mars с IP-адресом 192.168.0.1 в LAN подключен к ADSL-роутеру через Ethernet-интерфейс eth0. IP-адрес роутера — 10.0.0.138. Чтобы mars мог обмениваться информацией с ADSL-роутером, интерфейс eth0 конфигурируется статически (даже если ADSL-роутер поддерживает работу с DHCP, динамическая конфигурация с применением DHCP недопустима, так как в противном случае при каждом перезапуске компьютера в resolv.conf будут записываться данные DHCP ADSL-роутера!).

```
# в /etc/network/interfaces
...
# Статическое соединение с ADSL-роутером, то есть с Интернетом
auto eth0
iface eth0 inet static
  address 10.0.0.1
  netmask 255.255.255.0
  gateway 10.0.0.138
...
```

В файле /etc/resolv.conf содержится имя локального домена (sol) и IP-адрес локального сервера имен (то есть dnsmasq):

```
# /etc/resolv.conf
search       sol
nameserver   192.168.0.1
```

Еще остается настроить конфигурацию dnsmasq: теперь программа уже не может считывать адрес внешнего сервера имен (для обращения к Интернету) из resolv.conf. Наоборот, она должна игнорировать resolv.conf (параметр no-resolv) и связываться с внешним сервером имен, указанным после ключевого слова server:

```
# /etc/dnsmasq.conf
...
no-resolv
server=10.0.0.138
...
```

Итоговый файл

Если такой подробный рассказ о различных настройках dnsmasq вас немного запутал, вы можете просмотреть окончательный вариант файла dnsmasq.conf:

```
# /etc/dnsmasq.conf

# Интерфейс для LAN
interface=eth1,eth2

# Локальные хосты не предъявляются вышестоящему серверу имен
domain-needed
bogus-priv

# Доменное имя sol в LAN
local=/sol/
domain=sol
expand-hosts

# Dnsmasq для компьютера-шлюза (вышестоящий сервер имен = 10.0.0.138)
server=10.0.0.138
no-resolv

# Динамические адреса
dhcp-range=192.168.0.2,192.168.0.250,24h

# Статические адреса
dhcp-host=00:c0:ee:51:39:9f,pluto,192.168.0.254,infinite
```

Конфигурация для работы с несколькими интерфейсами

Нередко случается, что программа dnsmasq должна обслуживать несколько сетевых интерфейсов, и снабжать их IP-адресами из разных сетевых сегментов. В таком

случае нужно указать все интерфейсы с помощью interfaces. Кроме того, для каждого интерфейса потребуется задать собственную команду range:

```
# /etc/dnsmasq.conf
interface=eth1,eth2
dhcp-range=192.168.0.2,192.168.0.250,24h
dhcp-range=172.16.0.2,172.16.0.250,12h
...
```

В приведенном выше примере предполагается, что на шлюзе интерфейсу eth1 соответствует IP-адрес 192.168.0.1, интерфейсу eth2 — IP-адрес 172.16.0.1. При этом также становится ясно, какая команда dhcp-range относится к какому интерфейсу.

В отдельных случаях dnsmasq также требует, чтобы интерфейс был явно задан в команде dhcp-range:

```
# /etc/dnsmasq.conf
interface=eth1,eth2
dhcp-range=interface:eth1,192.168.0.2,192.168.0.250,24h
dhcp-range=interface:eth2,172.16.0.2,172.16.0.250,12h
...
```

В разделе NOTES на странице справки man dnsmasq указывается, что последний конфигурационный вариант требуется лишь изредка.

Журналирование

Программа dnsmasq автоматически заносит все динамически присвоенные IP-адреса в файл /var/lib/misc/dnsmasq.leases или /var/lib/dnsmasq/dnsmasq.leases. Статические адреса, напротив, не учитываются. В каждой записи этого файла также содержится MAC-адрес и хост-имена клиентов (если они известны). Таким образом, из этого файла очень удобно узнавать MAC-адреса новых клиентов.

Если dnsmasq работает не так, как вы хотели, вставьте в dnsmasq.conf ключевое слово log-queries. Теперь программа будет регистрировать все обращения к серверу имен в /var/log/syslog или /var/log/messages.

Клиентская конфигурация

В принципе сконфигурировать компьютер, который должен получать конфигурацию IP через DHCP, совсем не сложно. Все дистрибутивы Linux и современные версии Windows содержат соответствующий флажок в окне, используемом для конфигурации сети. Программа Network Manager, которая все шире используется в Linux, также самостоятельно распознает DHCP-данные LAN- или WLAN-соединения.

Единственный одновременно сложный и важный момент такой конфигурации — работа с хост-именами: как правило, клиент должен иметь фиксированное хост-имя, отправляемое на DHCP-сервер. Большинство клиентских DHCP-инструментов делают это по умолчанию. Исключениями являются Fedora и Red Hat. В этих дис-

трибутивах статически установленное хост-имя *не* передается обратно на DHCP-сервер! Если вы хотите, чтобы такая передача происходила, нужно отдельно задать имя к пересылке. Для этого запустите программу `system-config-network`, откройте командой Edit (Редактировать) окно со свойствами сетевого интерфейса (обычно это `eth0`) и укажите хост-имя в настройках DHCP. Такая настройка требуется и в тех случаях, когда вы устанавливаете связь с сетевым менеджером!

Как повторно считывать данные DHCP

Разумеется, чтобы испытать DHCP-сервер, не требуется каждый раз заново перезапускать клиенты. Вполне достаточно будет перезапустить сетевые функции. Если вы работаете с Network Manager, просто выберите в меню этой программы интерфейс, отвечающий за сетевые функции. Программа разорвет имеющееся соединение и установит его заново.

Если вы задаете сетевую конфигурацию, не пользуясь Network Manager, то заново запросите с помощью следующих команд информацию о DHCP:

```
root# /etc/init.d/networking restart  (Debian, Ubuntu)
root# service network restart         (Fedora, Red Hat, SUSE)
```

Затем убедитесь, что все функционирует. Для этого выполните команду `iconfig` и загляните в файл `/etc/resolv.conf`. Если вы работаете с KDE или Gnome, то после изменения хост-имени или других основополагающих сетевых параметров вам придется выйти из системы и снова в нее войти.

Даже в Windows можно заново считать данные DHCP без перезапуска. Откройте окно командной строки и выполните в нем следующую команду:

```
> ipconfig /renew              (Windows)
```

19.6. Интеграция WLAN в сеть

В том, чтобы дополнить обычную сеть функциями WLAN, нет ничего сложного. Простейшее решение — применить роутер ADSL-WLAN. Если вы настроили собственный интернет-шлюз так, как это было описано выше, то можете просто подключить к сетевому концентратору точку доступа к WLAN (рис. 19.2) или использовать вместо обычного концентратора локальной сети WLAN-роутер. В конфигурации шлюза ничего менять не надо. Нужно лишь настроить шифрование WLAN. В настоящее время лучше всего использовать метод шифрования WPA2 и пароль как можно длиннее.

К сожалению, против такого решения есть два аргумента, связанных с безопасностью.

- Все пользователи сети (независимо от того, подключаются они через LAN или WLAN) применяют одну и ту же подсеть 192.168.0.*. А задействовать некоторые сетевые службы (например, службы для работы с каталогами — NFS или Samba) только в безопасной сети LAN, но не использовать их в незащищенной сети WLAN невозможно.

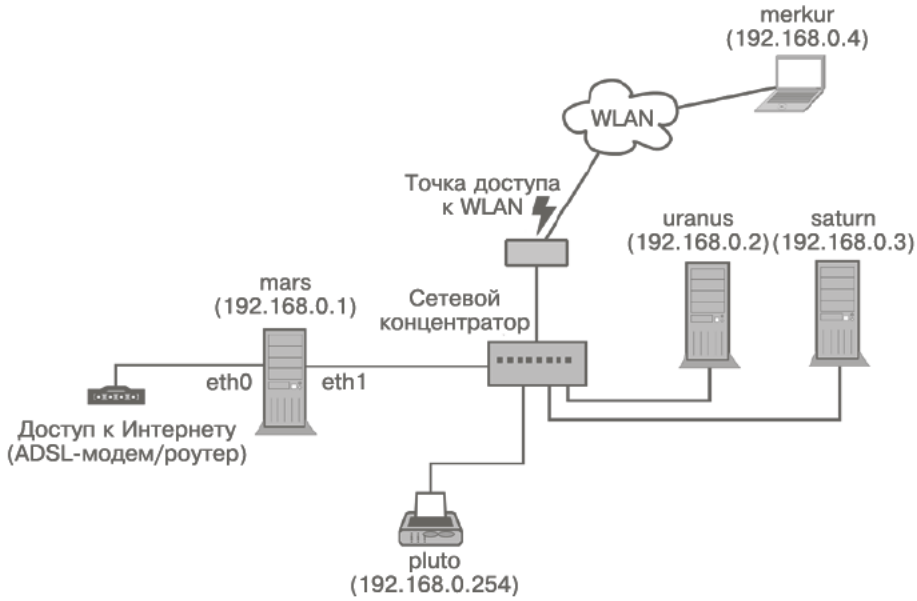


Рис. 19.2. Создание WLAN-соединения через сетевой концентратор

- Степень безопасности всей сети приравнивается к степени безопасности применяемой WLAN. При шифровании WLAN действует принцип наименьшего общего знаменателя: вы можете использовать метод шифрования WPA2, считающийся в наше время наиболее надежным, лишь в том случае, если все клиенты совместимы с WPA2.

Собственная подсеть WLAN с применением VPN. Можно подключить точку доступа WLAN не только к концентратору локальной сети, но и напрямую к компьютеру-шлюзу (рис. 19.3). Для этого на компьютере-шлюзе нужно установить третью сетевую карту.

Существенное преимущество второго варианта перед первым заключается в том, что во втором случае WLAN и LAN применяют два совершенно независимых сетевых адресных пространства, например 192.168.0.* для защищенной сети LAN и 172.16.0.* для незащищенной WLAN. При использовании второго метода непосредственное соединение клиентов LAN и WLAN становится невозможным, а это уже некоторое достижение в области безопасности. Обмен любыми данными между LAN и WLAN осуществляется через сервер, а на сервере установлен брандмауэр, который с точностью определяет, что разрешено, а что — нет. Кроме того, мы получаем возможность использовать в двух сетях разные сетевые службы — например, доступ к Интернету и SSH осуществляется и через LAN, и через WLAN, а доступ к Samba или MySQL возможен лишь через LAN.

Этот вариант становится особенно изящным, когда для защиты соединения WLAN между сетевым сервером и клиентом WLAN создается виртуальная частная сеть (Virtual Private Network, VPN): так обеспечивается так называемая бесшовная интеграция клиента WLAN и обычной сети без каких-либо компромиссов в области безопасности. О том, как создается VPN, рассказано в главе 27.

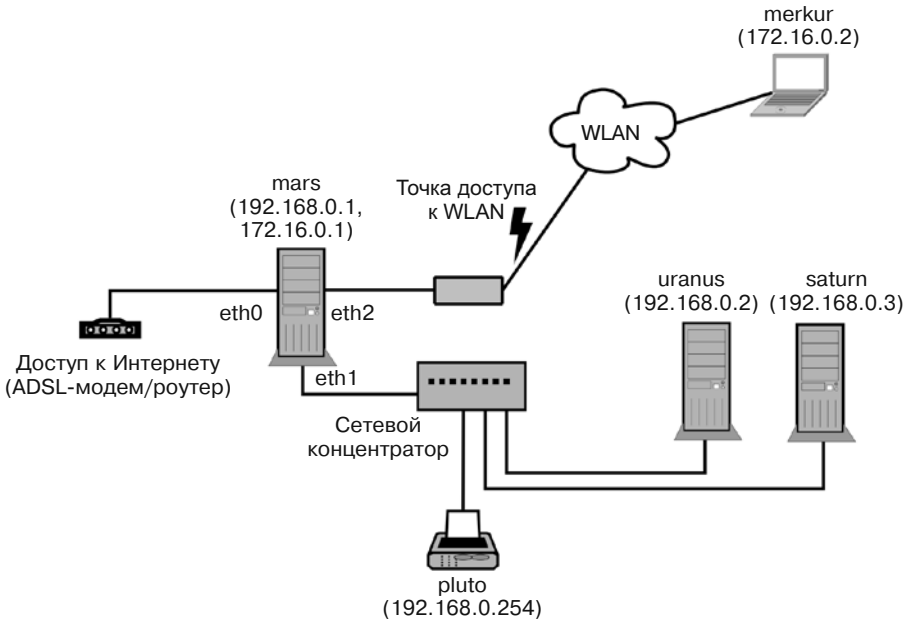


Рис. 19.3. Прямое подключение WLAN к интернет-шлюзу

20 Samba

С помощью Samba вы также можете предоставить для использования в сети файлы или каталоги с компьютеров с Linux. Пользователи, работающие с компьютерами Windows, Linux или Apple, смогут обращаться к этим файлам. Тонко дифференцированная система аутентификации и распределения прав определяет, кто имеет право на чтение и изменение файлов. Таким образом, Samba — это центральная узловая станция, предназначенная для обмена данными в локальной сети: на фирме, в организации или дома.

Название Samba — это вариация аббревиатуры SMB, которая, в свою очередь, обозначает протокол Server Message Block (блок сообщений сервера). Первые концепции, связанные с Samba, были разработаны в IBM, позже протокол дополняли многие фирмы, в первую очередь — Microsoft.

Часто Samba воспринимается как соединительное звено между мирами Windows и Linux. Но это слишком узкий подход: SMB часто применяется в средах, состоящих только из компьютеров с Linux или UNIX, так как упрощает работу: программы для просмотра файлов в Gnome и KDE по умолчанию поддерживают SMB, таким образом, с помощью CIFS вы можете подключать каталоги SMB непосредственно в дерево каталогов Linux и т. д. Если вы ищете UNIX-подобную альтернативу, которая работает не на основе Microsoft, то вам лучше всего использовать NFS. Но для применения NFS клиентские компьютеры должны иметь одинаковую пользовательскую конфигурацию, а это означает несовместимость с Windows.

В этом разделе будут рассмотрены основные функции Samba 3.6 с точки зрения сервера. Если вы хотите использовать многочисленные продвинутые функции, например аутентификацию через LDAP или использование Samba в качестве первичного контроллера домена, обратитесь к специальной литературе. Много полезной технической информации имеется на сайтах http://www.samba.org/ и https://help.ubuntu.com/community/SettingUpSamba.

20.1. Основы и глоссарий

Прежде чем перейти к конфигурации сервера Samba, разберемся в концепциях, лежащих в основе этого сервера, и узнаем, что означают основные связанные с ним термины.

NetBIOS. SMB работает на базе протокола NetBIOS (Network Basic Input/Output System). NetBIOS — это *сетевая базовая система ввода-вывода*. Ее разработку начала компания IBM. Со временем протокол NetBIOS был существенно обновлен. Первоначально NetBIOS состояла из трех служб.

- **Служба имен.** Этот метод обмена данными можно сравнить с DNS, используемым в UNIX/Linux. Управление именами может осуществляться с помощью сервера имен NetBIOS (NetBIOS Nameserver, NBNS) или децентрализованно. При использовании этого метода каждый запускающийся клиент посылает сообщение остальным клиентам сети и сообщает, под каким именем он начал работать.
- **Служба сеансов.** Этот механизм обмена информацией подобно TCP обеспечивает аккуратный обмен данными между двумя компьютерами — в форме пакетов. При этом проверяется целостность пакетов, и пакеты, которые содержат ошибки или потерялись, запрашиваются заново.
- **Служба датаграмм.** При использовании этого варианта службы сеансов система не проверяет, в порядке ли приходящие данные. Но у службы датаграмм есть и определенное достоинство — приходящие таким образом данные могут одновременно рассылаться на несколько компьютеров.

WINS. В Windows задачи NBNS решаются с помощью службы имен Интернета для Windows (Windows Internet Name Service, WINS). Сервер WINS можно настроить с помощью Samba или актуальной версии Windows (но не с Windows 9x). При этом, если в системе уже имеется сервер доменных имен, Samba может работать с ним.

Разрешение имен в Windows, в отличие от UNIX, может функционировать без применения специального сервера имен. Для этого необходимо рассылать всем компьютерам сети пакеты с датаграммами, поэтому такой метод становится все менее эффективным по мере того, как растут сети с Windows.

Имена компьютеров, используемые в NetBIOS и TCP/IP, не зависят друг от друга. Иными словами, возможен случай, когда компьютер, передавая данные по разным протоколам, будет при этом иметь разные имена. На практике таких случаев, разумеется, следует избегать. Различные имена не только вызывают путаницу, но и не позволяют использовать некоторые функции.

Просмотр сети и ресурсов. Откуда клиенту с Windows известно, какие еще компьютеры есть в сети? Для этого используется функция *просмотра сети и ресурсов*. Под этим понимается управление компьютерами, находящимися в сети. Чтобы таким управлением не нужно было заниматься каждому компьютеру, управление передается так называемой *главной системе просмотра ресурсов*. В больших сетях также работает одна или несколько резервных систем просмотра ресурсов. Кроме того, при необходимости Samba может выступать в качестве системы просмотра сети и ресурсов.

В системе не определяется жестко, какой именно компонент будет заниматься просмотром сети и ресурсов. Один из компьютеров в сети назначается для этой цели динамически — в зависимости от того, какие из компьютеров в настоящее время находятся в сети и какой из них наилучшим образом приспособлен для

данных целей (как правило, это компьютер с новейшей версией операционной системы). Это совсем не означает, что система автоматически выберет WINS-сервер, если он вообще есть в сети. Таким образом, функция просмотра сети и ресурсов реализуется очень децентрализованно.

К сожалению, ПК с Windows далеко не всегда удается синхронизировать список компьютеров (browsing list) и реальное состояние сети. Причина этого заключается в том, что клиенты управляют кэшем, в котором в целях снижения нагрузки на сеть сохраняется последнее рабочее состояние локальной системы. Кэш не всегда удается обновить сразу — бывает, что на это требуется определенное время. Иначе говоря, если вы не видите с компьютера с Windows другого компьютера под управлением Windows или Linux, но знаете, что искомый компьютер точно работает, то проблема заключается в несовершенстве функции просмотра сети (скорейшее решение такой проблемы, как это часто бывает при работе с Windows, — перезагрузка компьютера, который не удается найти).

Версии SMB. Когда речь идет о протоколе Server Message Block, обычно имеется в виду его первая версия. В Windows Vista и Windows Server 2008 по умолчанию применяется версия 2. В Windows 7 и Windows Server 2008 R2 — версия 2.1, а в Windows 8 и выше и Windows Server 2012 — версия 3.0. Разумеется, все версии Windows имеют обратную совместимость с SMB 1. Большинство нововведений, появившихся в SMB 2 и более поздних версиях, связаны с повышением эффективности работы системы.

Samba частично поддерживает SMB 2 начиная с версии 3.5 и практически полностью — с версии 3.6. (Единственное исключение составляют функции квотирования.) Но функции SMB 2 должны быть явно активизированы.

Права доступа и системы обеспечения безопасности

Название *«совместно используемые ресурсы»* (Shares) может обозначать и каталоги, и принтеры, которые предоставляются в пользование другим компьютерам через NetBIOS. Кроме того, мы будем говорить о каталогах и объектах. Существуют различные способы управления доступом, определяющие, какими ресурсами может пользоваться тот или иной компьютер.

- **Защита на уровне доступа к совместно используемым ресурсам.** При использовании простейшей формы управления правами доступа каждый каталог и принтер получает собственный пароль (разумеется, объекты могут предоставляться и без пароля). Такой метод все еще иногда применяется в некоторых небольших сетях. Самый очевидный недостаток — требуется очень много паролей: если каждому из 10 компьютеров должно быть предоставлено в пользование по три объекта, мы уже получаем 30 паролей. В больших сетях при применении такого метода достаточно скоро воцаряется хаос.
- **Защита рабочей группы.** Данный метод, являющийся более совершенной формой предыдущего, позволяет взаимный доступ к объектам лишь при условии, что два компьютера относятся к одной и той же рабочей группе. Это повышает

безопасность работы, но не кардинально: при отсутствии централизованного администрирования любой компьютер может «представиться» членом любой рабочей группы. Иначе говоря, защита на уровне доступа к совместно используемым ресурсам функционирует по децентрализованному одноранговому принципу (в противоположность методу клиент/сервер, применяемому на уровне пользователей и уровне доменов, который все сильнее тяготеет к централизации).

- **Защита на уровне пользователя.** Для реализации такой защиты пользователь должен применять для входа на клиентский компьютер логин и пароль. В результате, если пользователь захочет взять для работы определенные данные сервера Samba или другого ПК с Windows, основанием для доступа являются действующий логин и пароль этого пользователя. Кроме того, оба компьютера должны относиться к одной и той же рабочей группе.

  Таким образом, мы уже не присваиваем пароль каждому сетевому объекту. Теперь логин и пароль связаны с пользователем (со списком пользователей, перечисленных по именам, либо со всеми пользователями, входящими в определенную группу). Если в Samba задействуется защита на уровне пользователя, то вам потребуется отдельная база данных, в которой будут храниться имена пользователей, групповая отнесенность и пароли.

  Приведу пример: пользователь *X* работает на компьютере *A*. Чтобы *X* мог запрашивать данные с сервера Samba *S*, должны *X* и *A* быть зарегистрированы в системе *S* как пользователи (с одним и тем же именем и паролем). Теперь допустим, что компьютер *a* сломался. Пользователь *X* переходит на компьютер *B*. Чтобы пользователь *X* мог получить доступ к своим данным на *S* с компьютера *B*, на компьютере *B* потребуется создать новую учетную запись — опять же, с паролем.

  Если пользователь *X* решит изменить свой пароль, такое же изменение потребуется выполнить на сервере *S*, а следовательно, и на всех клиентских компьютерах (*A*, *B*, *C* и т. д.). Другими словами, при таком построении сети децентрализованное управление паролями и децентрализованная аутентификация остаются постоянными проблемами.

- **Защита на уровне доменов.** В версии Windows NT 4 в мире сетей Microsoft появились так называемые домены. Концепция управления доступом очень напоминает защиту на уровне пользователя. Различия касаются способов управления базой пользовательских данных и метода осуществления аутентификации.

  При входе в систему пользователь обращается к пользовательской базе данных, которая централизованно управляется сервером. Права доступа управляются с помощью так называемого регистрационного маркера. Входя в систему, клиент получает регистрационную информацию, которая остается действительной во всей сети до тех пор, пока пользователь не выйдет из сети. Он не замечает разницы, но внутри системы такое отличие становится фундаментальным и позволяет эксплуатировать сервер со значительно большей эффективностью.

  Для обеспечения защиты на уровне доменов необходимо, чтобы в сети был *первичный контроллер домена (Primary Domain Controller, PDC)*. В крупных сетях

кроме PDC может использоваться несколько *резервных контроллеров домена (Backup Domain Controller, BDC)*, чтобы вся система не останавливалась, если PDC выйдет из строя.

- **Active Directories.** Чтобы упростить управление крупными сетями, Microsoft дополнила механизм защиты на уровне домена функцией Active Directory. Для аутентификации (вход в систему, управление паролями и т. д.) теперь применяется облегченный протокол доступа к каталогу (LDAP). С его помощью сеть и ее домены можно построить иерархически. Кроме того, управление именами компьютеров в такой ситуации выполняется с помощью сервера доменных имен, как это обычно делается в Linux, и без применения WINS.

На клиентском уровне Samba поддерживает все пять перечисленных систем защиты. В настоящее время сервер Samba еще не может играть роль «Active Directory — домен — сервер». Эта возможность появится только в версии Samba 4. В данной книге мы рассмотрим только защиту на уровне пользователя. Таким образом, сервер Samba мы будем понимать как *изолированный* (standalone) сервер.

ПРИМЕЧАНИЕ

В конфигурационном файле Samba имеется несколько параметров, определяющих, какую систему защиты будет использовать Samba. Важнейший из этих параметров — security. Обратите внимание на то, что его настройки могут не совпадать с перечисленными выше вариантами один к одному! По умолчанию действует настройка security=user, то есть защита на уровне пользователя. Но эта настройка сохраняется и при конфигурации PDC. Остальные параметры важны для того, чтобы Samba производила аутентификацию и при входе пользователей на компьютеры с Windows.

Централизованная или децентрализованная топология сервера

Если не учитывать систему защиты, существует две принципиально различные стратегии обмена данными между компьютерами в сети с применением Samba.

- **Централизованная топология.** Центральный сервер Samba предоставляет всем пользователям доступ к каталогам. Задача администратора — настроить права доступа к отдельным каталогам так, чтобы в системе имелись и закрытые (частные) каталоги отдельных пользователей, и относительно открытые каталоги для обмена данными в пользовательских группах.

  Преимущества такого варианта конфигурации заключаются в том, что данные централизованно сохраняются на сервере и отдельные пользователи не должны сами заниматься созданием сетевых каталогов. У этой конфигурации есть и недостатки: система получается недостаточно гибкой, любые изменения вносятся администратором. Кроме того, выход сервера из строя чреват катастрофическими последствиями для всей сети.

- **Децентрализованная топология.** При использовании такого варианта каждый компьютер, который должен предоставить данные для совместной работы в сети, делает это сам. И в Windows, и в Linux (точнее говоря, в Gnome и KDE) система помогает в этом пользователю относительно простыми диалоговыми окнами.

Преимущество такого варианта конфигурации — децентрализованный подход, при котором каждый отвечает сам за себя и не нуждается в помощи администратора. По мере роста сети конфигурация, разумеется, становится все более запутанной, централизованное резервное копирование оказывается практически неосуществимым.

Я придерживаюсь такого мнения, что на предприятии более целесообразно применять централизованную топологию сети. Если же вам нужно просто скопировать пару файлов с одного компьютера на другой в частной сети, то, конечно же, вполне хватит и временной децентрализованной конфигурации, настраиваемой через диалоговое окно общего доступа из Gnome или KDE. Основная проблема заключается в том, что такие диалоговые окна Gnome или KDE недоработаны и плохо функционируют.

20.2. Samba: базовая конфигурация и ввод в эксплуатацию

Установка. Во многих дистрибутивах для сервера и для клиента применяются отдельные пакеты. Обычно клиентские пакеты устанавливаются по умолчанию, поэтому доступ к сетевым каталогам должен функционировать с ходу. Но если вы хотите предоставить в общее пользование сами сетевые каталоги, то вам не обойтись без серверных функций, которые в большинстве дистрибутивов упакованы в пакете `samba`.

В некоторых дистрибутивах также предоставляются пакеты Samba 4. Но эти пакеты ориентированы исключительно на разработчиков, которые сами хотят протестировать функции Samba. В середине 2012 года Samba 4 еще не годилась для решения практических рабочих задач (хотя создается впечатление, что версия уже почти готова).

Запуск. Службы Samba предоставляются с помощью двух фоновых процессов:

- `nmbd` — применяется для внутрисистемного управления и как сервер имен, а также отвечает за просмотр сети и ресурсов; его можно использовать и в качестве главной системы просмотра ресурсов или как WINS-сервер;
- `smbd` — это клиентский интерфейс, предоставляющий клиентам доступ к каталогам, принтерам и к текущему списку компьютеров.

Оба процесса запускаются системой Init-V. Названия сценариев Init-V зависят от применяемого дистрибутива. Если в вашем дистрибутиве по умолчанию не предусмотрен запуск Samba сразу же после установки, то обратитесь к разделу 8.5 — там даются советы, как запускать сценарии и конфигурировать автоматический запуск.

- Debian — `/etc/init.d/samba` запускает оба процесса.
- Fedora — Systemd запускает Samba.
- RHEL, SUSE — `/etc/init.d/smb` запускает `smbd`, `/etc/init.d/nmb` запускает `nmbd`.
- Ubuntu — Upstart запускает Samba (`/etc/init/nmbd.conf` и `/etc/init/smbd.conf`).

Smb.conf. Основной конфигурационный файл Samba называется `/etc/samba/smb.conf`. Файл состоит из глобального раздела для основных настроек (`[global]`), а также из любого количества других разделов, предназначенных для предоставления ресурсов (каталогов, принтеров и т. д.) в совместное пользование. Каждый раздел начинается с `[имя_ресурса]`. Комментарии могут вводиться символом # или ;. Нельзя ставить комментарий сразу после настройки параметра, то есть каждый комментарий должен начинаться с новой строки.

Далее с небольшими сокращениями показан фрагмент стандартной конфигурации Samba в Ubuntu. В остальных дистрибутивах этот файл (без учета комментариев) часто еще короче, поскольку настройки, задаваемые по умолчанию, в нем отдельно не указываются.

В следующем примере опущены разделы `[printers]` и `[print$]`, обеспечивающие доступ к принтеру и его драйверам.

```
# Файл /etc/samba/smb.conf в Ubuntu 12.10
[global]
workgroup               = WORKGROUP
server string           = %h server (Samba, Ubuntu)
dns proxy               = no
log file                = /var/log/samba/log.%m
max log size            = 1000
syslog                  = 0
panic action            = /usr/share/samba/panic-action %d
encrypt passwords       = true
passdb backend          = tdbsam
obey pam restrinctions  = yes
unix password sync      = yes
passwd program          = /usr/bin/passwd %u
passwd chat             = ...
pam password change     = yes
map to guest            = bad user
usershare allow guests  = yes
```

Удаление комментариев из smb.conf. В некоторых дистрибутивах файл `smb.conf` применяется в качестве документации. Получающийся в итоге конфигурационный файл крайне длинный и сложный. Чтобы упростить его, используются три следующие команды:

```
root# cd /etc/samba
root# cp smb.conf smb.conf.org
root# grep -Ev '^#|^;|^$' smb.conf.orig > smb.conf
```

Идентификация сервера. С помощью `workgroup` определяется имя рабочей группы. Вероятно, это первая настройка, которую вам придется изменить, чтобы вписать здесь имя вашей рабочей группы, в которой будет действовать Samba.

Настройка `server string` указывает, под каким именем будет идентифицироваться сервер. Здесь `%h` заменяется хост-именем.

WINS. Благодаря настройке `dns proxy=no` Samba, работающая как WINS-сервер, не будет обращаться к DNS для разрешения хост-имен Windows. Если в вашей сети LAN будет локальный сервер имен, установите для этого параметра значение `yes`.

Стандартное значение no пригодится вам только в тех случаях, когда в сети нет локального сервера имен, к которому можно быстро обратиться.

Журналирование. Параметры log file, max log size и syslog определяют, какие данные будет регистрировать Samba и где. Журналирование подробно рассматривается в одноименном подразделе далее.

При аварийном завершении работы Samba выполняется сценарий panic-action. Он отсылает администратору электронное сообщение, в котором содержится информация о возникших ошибках. Если на сервере не установлена почтовая система, то panic-action будет бездействовать.

Пароли. Настройка passdb backend указывает, как Samba должна работать с паролями. На выбор предлагаются варианты smbpasswd (обычный текстовый файл), tdbsam (TDB, относительно простая база данных) или ldap-sam (LDAP). Обычно в небольших или средних сетях (примерно до 250 клиентов) лучше всего использовать значение tdbsam. Систему smbpasswd, которая раньше была достаточно популярна, в настоящее время применять не рекомендуется, так как в ней нельзя сохранять расширенные атрибуты, используемые в Windows NT 4 и выше (SAM Extended Controls).

Ключевые слова unix password sync, passwd chat и pam password change указывают, должна ли Samba сравнивать свои пароли с паролями Linux и, если должна, то как именно.

Гости. С помощью map to guest и usershare guest определяется, как Samba должна поступать с пользователями, не прошедшими аутентификацию, то есть с пользователями, которые пытаются войти в систему под недействительным именем или паролем.

Модель защиты. Возможно, вы заметили, что в последнем листинге отсутствует указание на модель защиты. По умолчанию в Samba, а значит, и в базовой конфигурации Ubuntu действует защита на уровне пользователя (security=user). Если вы желаете применить иную модель защиты, ее нужно настроить как значение параметра security.

Стандарты Samba. В Debian и Ubuntu в файле smb.conf содержатся некоторые команды, отсутствующие в предыдущем листинге, поскольку они избыточны. Например, пароли обязательно шифруются на протяжении уже многих лет, поэтому encrypt passwords=true просто документирует стандартную настройку. Вас может запутать настройка obey pam restrictions=yes: она влияет на управление паролями лишь в том случае, когда пароли не шифруются. Поскольку в нашем случае пароли шифруются, эта настройка игнорируется.

Далее нам предстоит познакомиться еще со многими параметрами Samba, но, конечно же, не со всеми. Подробная информация обо всех возможностях настройки дается в man smb.conf.

SMB 2. Версии Samba 3.5 и 3.6 поддерживают новую версию протокола SMB 2. Но соответствующие дополнения по умолчанию неактивны, и их нужно явно активизировать. Для этого добавьте в глобальный раздел файла smb.conf такую команду:

```
# в /etc/samba/smb.conf
[global]
    ...
    max protocol = SMB2
```

Изменения конфигурации, статус

Чтобы изменения, внесенные в smb.conf, вступили в силу, Samba должна заново считать конфигурационные файлы:

```
root# service samba reload  (Debian)
root# service smbd reload   (Fedora, openSUSE, Ubuntu)
```

Testparam. Если вы вносите в smb.conf значительные изменения, проверьте с помощью команды testparam, нет ли в коде синтаксических ошибок:

```
root# testparm
Load smb config files from /etc/samba/smb.conf
Processing section "[printers]"
Processing section "[print$]"
Loaded services file OK.
Server role: ROLE_STANDALONE
Press enter to see a dump of your service definitions <Return>
[global]
  server string = %h server (Samba, Ubuntu)
  ...
```

Если выполнить testparam с параметром -v, то команда выдаст очень длинный список всевозможных способов настройки smb.conf. Этот список может вам пригодиться в том случае, когда вы не знаете точно, какие настройки действуют по умолчанию (то есть какие параметры функционируют без вашего вмешательства в систему).

Чтобы узнать актуальное состояние сервера Samba, используйте smbstatus. Эта команда также выводит список всех соединений, которые активны в настоящий момент.

Защита Samba

Брандмауэр. В Fedora, SUSE и RHEL Samba по умолчанию блокируется брандмауэром. Чтобы можно было использовать сетевые службы, нужно разблокировать специфичные для Samba и Windows TCP-порты 135, 139 и 445, а также UDP-порты 137 и 138. Это делается на интерфейсе, стоящим на входе в локальную сеть.

Но полностью отключать брандмауэр не рекомендуется! Со стороны Интернета любой сервер Samba обязательно должен быть защищен брандмауэром! Если этого не обеспечить, то предоставленные в совместное использование каталоги будут доступны где угодно. Правда, они могут быть дополнительно защищены пользовательскими паролями Samba.

Интерфейсы. Независимо от конфигурации Samba, не теряйте бдительности. Специально задайте с помощью interfaces, через какие сетевые интерфейсы сервер должен обмениваться информацией с остальными компьютерами. Интерфейсы указываются не по именам, а по адресным пространствам, используемым этими интерфейсами. Не забудьте о localhost, иначе на сервере не будут работать такие инструменты администрирования, как smbclient или SWAT. Этот параметр имеет значение лишь в том случае, если ваш компьютер оборудован несколькими сете-

выми интерфейсами (на многих компьютерах имеются интерфейсы не только для физических сетевых адаптеров, но и для виртуальных сетевых адаптеров различных программ виртуализации!). По умолчанию Samba обслуживает *все* сетевые интерфейсы.

Чтобы настройки, сделанные в `interfaces`, вступили в силу, вам потребуется специально активизировать настройку `bind interfaces only`.

Хосты. Далее вы можете перечислить в `hosts allow` те компьютеры, которые имеют право обмениваться информацией с Samba. Хост-имена, IP-адреса и IP-адресные пространства в этом списке отделяются друг от друга пробелами. В `hosts allow` можно сделать еще более точную выборку, чем в `interfaces`. Дополнительно в `hosts deny` можно указать отдельные хосты и адреса, которые нельзя использовать Samba.

Гости. Наконец, настройка `map to guest` запрещает всем пользователям, которые не прошли аутентификацию на сервере, любой доступ в сеть. В некоторых случаях может быть очень целесообразно создать для гостей отдельные каталоги, файлы в которых можно читать и даже изменять без аутентификации; однако если этого не требуется, нужно изначально закрыть гостям доступ в сеть.

```
# /etc/samba/smb.cnf
[global]
  ...
  bind interfaces only= yes
  interfaces          = 192.168.0.0/24 localhost
  hosts allow         = clientA clientB clientC
  map to guest        = never
```

Журналирование

Службы Samba `smbd` и `nmbd` регистрируют глобальные события в файлах `/var/log/samba/log.smbd` и `log.nmbd`. Ни название, ни местоположение этих файлов регистрации нельзя изменить с помощью `smb.conf`.

Параметр `log file` в глобальном разделе `smb.conf` указывает, где должны сохраняться клиентские уведомления. Благодаря предварительной настройке `/var/log/samba/log.%m` для каждого клиента, который обращается к Samba, создается свой файл регистрации с именем `log.hostname`. Параметр `max log size=1000` ограничивает максимальный размер 1000 Кбайт. Если файл регистрации превышает этот размер, то Samba переименовывает его — `name.old`. Параметр `syslog=0` не означает, что `syslog` не применяется, а говорит о том, что в `/var/log/syslog` должны регистрироваться только сообщения об ошибках. Могут также применяться значения `1`, `2`, `3` и т. д., если вы хотите регистрировать предупреждения, замечания и сообщения об отладке.

Logrotate. Будьте осторожны, применяя программу `logrotate` (см. раздел 11.7): согласно настройке, действующей в Debian, SUSE и Ubuntu по умолчанию, эта программа раз в неделю архивирует файлы `log.smbd` и `log.nmbd` и одновременно удаляет все архивные версии, которым больше двух месяцев. Однако `logrotate` игнорирует клиентские файлы регистрации `log.hostname`, которые прирастают значительно быстрее. Аналогичный конфигурационный файл в Fedora и Red Hat

построен более рационально, и в нем учитываются *все* файлы регистрации, находящиеся в /var/log/samba:

```
# /etc/logrotate.d/samba в Fedora и Red Hat
/var/log/samba/* {
  notifempty
  olddir /var/log/samba/old
  missingok
  sharedscripts
  copytruncate
}
```

В качестве альтернативы вы можете использовать в smb.conf настройку log file = /var/log/samba/log.smbd. Таким образом, smbd будет регистрировать в одном и том же файле и глобальные, и клиентские уведомления. Если вы не ставите своей задачей отыскать ошибки в конфигурации Samba, лучше всего выбрать именно это решение.

Сетевая конфигурация с помощью SWAT

Чтобы не изменять конфигурационный файл Samba smb.conf в текстовом редакторе, можете применить Samba Web Administration Tool (SWAT). Этапы установки SWAT и введения его в эксплуатацию различаются в зависимости от дистрибутива. В любом случае перед установкой сохраните резервную копию smb.conf.

ПРИМЕЧАНИЕ

Среди Samba-профессионалов у SWAT не самая хорошая репутация. Этот инструмент лишь немного помогает при конфигурации. Если вам удается выполнить нужную конфигурацию с помощью SWAT, вы с тем же успехом можете изменять файл самостоятельно — без тех ограничений, с которыми связана работа в SWAT.

Если вы сохраняете изменения конфигурации, то SWAT удаляет все имеющиеся комментарии из smb.conf, изменяет порядок уже имевшихся записей и «на свое усмотрение» изменяет некоторые настройки. Поэтому рекомендую уже на этапе установки создать резервную копию smb.conf.

Debian, Ubuntu. SWAT находится в одноименном пакете, который необходимо специально установить. Вместе с ним инсталлируется пакет openbsd-inetd. Чтобы он учитывал изменения, автоматически вносимые в конфигурационный файл /etc/inetd.conf, перезапустите openbsd-inetd:

```
root# service openbsd-inetd restart
```

Для работы SWAT необходимо, чтобы у учетной записи root был пароль. В Ubuntu по умолчанию это не так. Выполните sudo passwd root и задайте надежный пароль администратора!

Fedora, Red Hat. SWAT находится в пакете samba-swat, который устанавливается дополнительно. Одновременно с ним инсталлируется пакет xinetd. Чтобы активизировать SWAT, замените в /etc/xinet.d/swat строку disable=yes строкой disable=no и запустите xinetd.

```
root# /etc/init.d/xinetd start
root# chkconfig --add xinetd
```

SUSE. В SUSE SWAT содержится в пакете `samba`. SWAT нужно специально активизировать, заменив в `/etc/xinet.d/swat` строку `disable=yes` строкой `disable=no`. Кроме того, не забудьте запустить `xinetd`:

```
root# insserv xinetd
root# service xinetd start
```

Применение. SWAT использует миниатюрный веб-сервер для обмена информацией через порт 901 (установка Apache не требуется!). Чтобы начать работу со SWAT, введите в адресной строке браузера адрес `http://localhost:901` или `http://127.0.0.1:901`.

После этого появляется окно для входа в систему, в котором вы должны представиться как root. Остальные пользователи не имеют достаточных прав для изменения конфигурационного файла Samba и могут только узнать статус Samba, а также изменить собственный пароль для Samba. В эпоху Web 2.0 пользовательский интерфейс SWAT может показаться несколько старомодным (рис. 20.1).

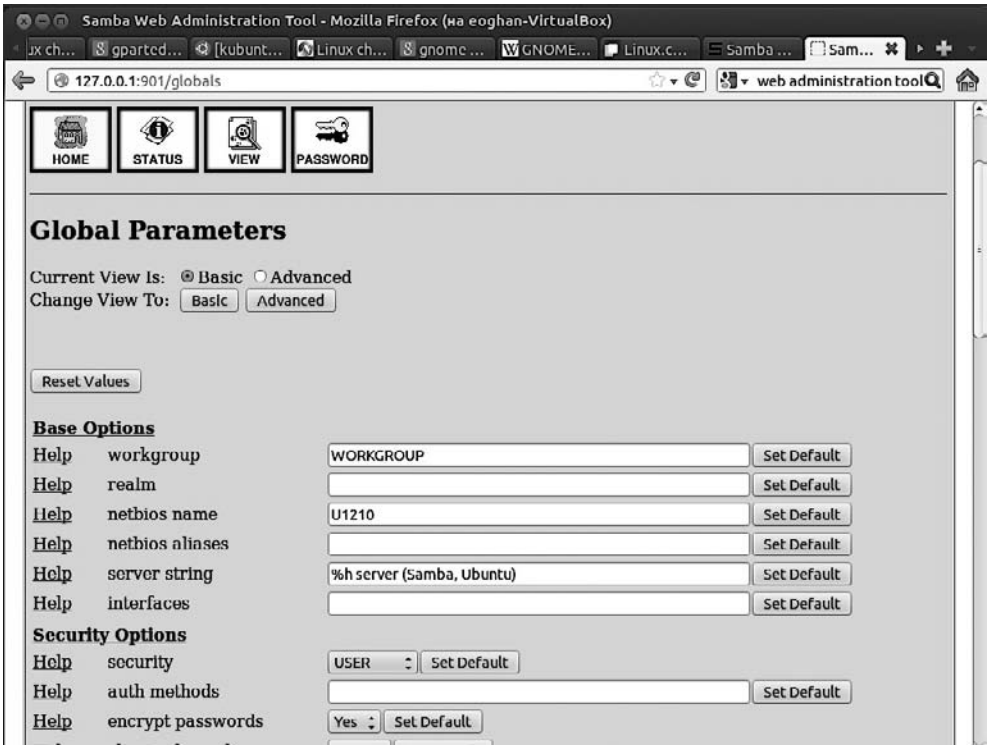


Рис. 20.1. Конфигурация Samba с помощью SWAT

Защита. При входе в систему имя пользователя и пароль передаются в незашифрованном виде, поэтому SWAT из соображений безопасности следует применять только на локальном компьютере или в локальной сети. В Debian и Ubuntu для этого необходимо изменить файлы `/etc/hosts.deny` и `hosts.allow` так, как это

описано в разделе 26.2 (в следующем примере показан случай с локальной сетью, работающей в адресном пространстве 192.168.0.*):

```
# Файл /etc/hosts.allow
swat : 192.168.0.0/24

# Файл /etc/hosts.deny
swat : ALL
```

В Fedora, Red Hat и SUSE этого выполнять не требуется, так как на основании строки `only_from=127.0.0.1` в `/etc/xinet.d/swat` соединение со SWAT может устанавливать только localhost.

20.3. Управление паролями

По умолчанию в Samba действует настройка `security=user`, то есть защита на уровне пользователя. Например, чтобы пользователь peter мог работать сетевым каталогом, должны выполняться следующие условия.

- На сервере должна быть учетная запись Linux с именем peter. Она требуется для управления правами доступа. Samba обращается к правам доступа, действующим в Linux, чтобы определить, какие пользователи имеют право на чтение и изменение файлов.

  Учетная запись Linux должна быть неактивна. Часто из соображений безопасности бывает нужно задать для учетной записи недействительный пароль. Таким образом, исключается вероятность, что пользователь Samba войдет на сервер.
- peter и его пароль должны содержаться в базе данных пользователей Samba. По техническим причинам учетные записи Samba (имя пользователя, пароль и другие данные) управляются независимо от учетных записей Linux. Пароль учетной записи Linux не имеет для Samba никакого значения.
- В `smb.conf` должны быть указаны сетевые каталоги, которые может использовать peter.

Пароли Samba

Прежде чем пользователь локальной сети сможет получить доступ к каталогу, он должен пройти идентификацию на сервере Samba. При установлении соединения с клиентом Windows для этого сообщаются логин Windows и зашифрованная последовательность символов — пароль. На клиентских компьютерах, использующих Linux, эти данные нельзя получить, зная учетную запись Linux, поэтому обычно на экран выводится окно для ввода имени пользователя и пароля Samba. Затем, как и с клиентами Windows, процесс продолжается: пароль на сервер Samba отправляется в зашифрованном виде, а не открытым текстом.

Из соображений безопасности пароль нельзя восстановить, имея зашифрованную последовательность символов, в виде которой он передается на сервер. Эта последовательность на сервере Samba сравнивается с другой последовательностью,

зашифрованной аналогичным образом. Если обе последовательности символов совпадают, значит, совпадают и пароли.

Внутренний интерфейс базы данных smbpasswd. Для работы серверу Samba требуется база данных, в которой хранятся имена пользователей и пароли, с помощью которой осуществляется аутентификация. Раньше для этих целей зачастую применялся обычный текстовый файл (`passdb backend=smbpasswd`). В SUSE этот внутренний интерфейс по-прежнему используется и пароли сохраняются в файле `/etc/samba/smbpasswd`.

Внутренний интерфейс TDB. В большинстве других дистрибутивов для небольших сетей применяется интерфейс TDB (`passdb backend=tdbsam`). В более крупных сетях данными учетных записей лучше управлять через LDAP, но этот вопрос мы отдельно обсуждать не будем.

TDB означает Trivial Database (Простая база данных) и представляет собой двоичный формат для сохранения наборов данных. Существенное преимущество этого формата по сравнению с обычными файлами `smbpasswd` заключается в том, что кроме логина и пароля в нем можно сохранять и другие данные, в частности атрибуты. К таким данным относятся, например, SAM Extended Controls, улучшающие совместимость с современными версиями Windows. От дистрибутива зависит, где именно физически сохраняются пароли:

- Debian, Ubuntu — `/var/lib/samba/passdb.tdb`;
- Fedora, Red Hat —`/var/lib/samba/secret/passdb.tdb`;
- SUSE — `/etc/samba/passdb.tdb`.

При необходимости вы можете указать в `smb.conf` другой файл с помощью настройки `passdb backend=tdbsam:`*`имя_файла`*. Команда `smbpasswd` создает новую учетную запись Samba или изменяет ее пароль; `pdbedit` обеспечивает доступ ко всей информации, касающейся учетной записи: таким образом, администраторы могут создавать списки всех пользователей Samba (`pdbedit -L -v`), указывать для каждой учетной записи специальные атрибуты и т. д. Обе команды интерпретируют `smb.conf` и работают со всеми системами паролей (то есть `smbpasswd`, `tdbsam` и `ldapsam`).

Smbpasswd. Итак, чтобы пользователь Петер мог работать с каталогом Samba, вы как системный администратор Linux должны создать в Samba учетную запись peter. Для этого используйте команду `smbpasswd`. В качестве пароля укажите последовательность символов, которая является паролем пользователя peter в Windows. Чтобы изменить уже имеющийся пароль Samba, примените `smbpasswd` без параметра `-a`.

```
root# smbpasswd -a peter
New SMB password: ******
Retype new SMB password: ******
Added user peter.
Password changed for user peter.
```

Обратите внимание — чтобы команда `sbmpasswd` работала, в локальной сети также должен существовать пользователь Linux peter (файл `/etc/passwd`). При необходимости новые пользователи Linux добавляются с помощью команд `useradd` или `adduser`.

Синхронизация паролей Samba и Linux

Обычно при работе в сети бывает необходимо, чтобы пользователи каталогов Samba могли регистрироваться прямо на сервере, например для работы с SSH и обработки своих файлов с применением команд Linux. В таких случаях, разумеется, пароли Samba и Linux обязательно должны совпадать. Наконец, было бы очень неудобно вносить каждое изменение пароля по нескольку раз (в самых сложных случаях трижды: для клиента Windows, сервера Samba и учетной записи Linux).

К сожалению, выполнить синхронизацию паролей не так просто, поскольку для шифрования паролей Samba используется иной алгоритм, нежели для паролей Linux. Поэтому управление паролями Samba и Linux происходит отдельно. (Хотя алгоритмы и разные, между ними есть одна общая черта: сохраняемые последовательности символов позволяют проконтролировать пароль, но не дают возможность его реконструировать. По этой причине преобразовать или конвертировать сохраненные пароли из одной системы в другую невозможно.)

Наиболее популярное решение этой проблемы заключается в том, чтобы при каждом вызове `smbpasswd` со стороны клиента (для изменения пароля) параллельно с этим изменялся и пароль для Linux. В Ubuntu `smbpasswd` уже содержит необходимые для этого настройки:

```
[global]
...
  unix password sync  = yes
  pam password change = yes
  passwd chat         = *Enter\snew\s*\spassword:* %n\n
                        *Retype\snew\s*\spassword:* %n\n
                        *password\supdated\ssuccessfully*
```

Команда `unix password sync=yes` активизирует синхронизацию. При этом по причине использования `pam password change` применяется PAM. PAM — это подключаемые модули аутентификации, представляет собой собрание библиотек для администрирования паролей. Параметр `passwd program`, который требовался ранее и задавал путь к программе `passwd`, в PAM не имеет значения и игнорируется. Параметр `passwd chat` описывает обмен данными между Samba и PAM. Эта последовательность символов в предыдущем листинге была разделена на три строки, но в `smb.conf` она указывается одной строкой.

К сожалению, синхронизация связана со множеством ограничений.

- Команда `smbpasswd` выполняется конкретным пользователем, а не администратором! Причина: если `smbpasswd` выполняет root, он переходит к работе непосредственно с базой данных Samba (этот механизм действует и в том случае, когда сервер Samba не работает). Если же командой управляет обычный пользователь, эта команда связывается с сервером Samba, который и выполняет нужную нам задачу, в том числе синхронизирует пароль.
- Команда `smbpasswd` принимает даже самые плохие пароли (например, состоящие из пробела или только из одной буквы). Система паролей Linux или PAM, напротив, требует хотя бы минимально качественного пароля и не принимает совсем простых вариантов. В результате пароль Samba может измениться, а па-

роль Linux останется прежним. С этого момента синхронизация паролей закончится и любая новая попытка изменить пароль Linux потерпит неудачу. Чтобы снова синхронизировать пароли, администратору потребуется заново настроить пароли Samba и Linux для конкретного пользователя.

- Синхронизация с применением `unix password sync` функционирует лишь в одном направлении: от Samba к Linux. Если же пользователь изменит свой пароль для Linux с помощью `passwd`, пароль Samba останется прежним (в таком случае помогает библиотека `libpam-smbpass`, см. ниже).

Из личного опыта я не рекомендую выполнять описанную выше синхронизацию паролей. При этом очень легко допустить ошибку и создать гораздо больше проблем, чем вы можете решить, применив такую синхронизацию. Используйте настройку `unix password sync=no`.

Libpam-smbpass. Для синхронизации паролей по направлению от Linux к Samba в Ubuntu применяется библиотека `libpam-smbpass`. При установке одноименного пакета конфигурационные файлы PAM (подробнее о них — в подразделе «Подключаемые модули аутентификации (PAM)» раздела 11.4) изменяются так, что после каждого успешного входа в систему либо после изменения пароля Linux соответствующий пароль Samba также создается или обновляется. (Предполагается, что вы не активировали автоматический вход в вашу настольную систему!)

Библиотека `libpam-smbpass` очень удобна, прежде всего, в тех случаях, когда несколько пользователей Linux хотят обмениваться файлами по Samba. Библиотека `libpam-smbpass` гарантирует, что пароль Samba в любом случае будет идентичен соответствующему ему паролю Linux.

ПРИМЕЧАНИЕ

Не путайте библиотеку libpam-smbpass с [lib]pam_smb! Модуль pam_smb обеспечивает аутентификацию (вход в Linux) на сервере Samba или Windows и никак не связан с синхронизацией паролей.

Соотнесение пользователей Linux и Windows

В Windows именем пользователя может служить практически любая последовательность символов (до 128 знаков). в Linux длина имени пользователя, напротив, составляет всего 32 знака — без специальных символов и пробелов. Если пользователь Windows применяет имя, которое не может быть именем пользователя в Linux, две этих последовательности необходимо соотнести через специальный файл. Название данного файла указывается в `smb.conf` с помощью параметра `username map`:

```
# /etc/samba/smb.conf
[global]
    ...
    username map = /etc/samba/smbusers
```

В каждой строке файла `smbusers` сначала указывается имя пользователя в Linux, затем символ = и одно или несколько имен пользователя в Windows. Имена,

содержащие пробелы, задаются в кавычках. Файл можно использовать и в том случае, если нескольким пользователям Windows соответствует один пользователь Linux.

```
# /etc/samba/smbusers
peter = "Peter Mayer"
...
```

ВНИМАНИЕ

Файл /etc/samba/smbusers может вывести из строя систему безопасности Samba или Linux, если имя root будет присвоено другому пользователю, кроме администратора! Обратите внимание, что этот файл разрешено изменять только пользователю root:

```
root# chmod 644 /etc/samba/smbusers
```

Все вместе

Предположим, в вашей локальной сети есть компьютер с Windows, на котором работает Петер Майер, и на этом компьютере используется логин Peter Mayer. На сервере Linux, где установлена Samba, есть учетная запись peter и программа `smbusers`, которая соотносит друг с другом Peter Mayer и peter. При таких условиях мы получим следующие сочетания логина и пароля.

- Логин в Windows — Peter Mayer и пароль Windows. Пароль Windows используется при работе в пределах локального компьютера с Windows. Петер может изменить свой пароль в Windows.
- Логин на сервере (Linux) — peter и пароль Linux. Пароль Linux применяется для входа непосредственно на сервер, если учетная запись Linux активна и не заблокирована. Петер может изменить свой пароль, войдя на сервер через SSH и применив команду `passwd`.
- Доступ к сетевым каталогам — Peter Mayer или peter и пароль для Samba. Пароль Samba используется для работы с сетевыми каталогами. Он должен совпадать с паролем Windows. Если это не так, то когда Петер захочет получить доступ к сетевому каталогу, в Windows откроется окно для входа в систему.

  Петер может изменить пароль с помощью команды `smbpasswd` после входа в SSH. Если на сервере установлен SWAT, то Петер также может изменить свой пароль в браузере. В любом случае в SWAT Петер должен входить с тем паролем, который используется для *Linux*, и только после этого он сможет изменить пароль, применяемый в *Samba*. Следует признать, что схема довольно запутанная. Если учетная запись Linux заблокирована, то Петер никак не может сам изменить свой пароль для Samba!

Таким образом, только те пользователи, которые отлично подкованы технически, могут сами изменять все три своих пароля — и лишь при условии, что соответствующая учетная запись Linux активна.

Для всех остальных пользователей действует правило: однажды заданные пароли никогда изменяться не будут. С точки зрения безопасности это, конечно же, совсем не хорошо.

20.4. Сетевые каталоги

В предыдущем разделе мы рассмотрели, какие предпосылки должны выполняться, чтобы пользователь мог войти в Samba. Если сказать кратко, то у пользователя должны быть учетная запись в Linux и пароль к Samba. Однако пока остается открытым вопрос, какие ресурсы увидит и сможет применить пришедший пользователь. За это отвечают разделы `[имя_ресурса]` в `smb.conf`.

Пользовательские каталоги

Определение каталога, к которому будет иметь доступ конкретный пользователь, выглядит так:

```
# в /etc/samba/smb.conf
...
[directory1]
  user      = peter
  path      = /data/dir1
  writeable = yes
```

Такая настройка позволяет пользователю peter, а также всем другим пользователям, которые в соответствии с `smbusers` работают с данной учетной записью, читать каталог `/data/dir1` и вносить в него изменения. В файловом менеджере этот ресурс называется `directory1` (то есть последовательность символов, указанная в квадратных скобках).

Значения ключевых слов понятны: `user` — это имя пользователя. Вместо `user` также могут использоваться ключевые слова `users` или `username`. Можно указать несколько имен пользователя, разделяя их запятыми.

Слово `path` говорит о том, какой каталог сервера предоставляется в совместное использование. Если `path` специально не указать, то Samba по умолчанию предоставляет в совместное использование домашний каталог заданного пользователя. С помощью `writeable=yes` мы разрешаем вносить в каталог изменения. Без этого параметра пользователь имеет доступ только для чтения.

В принципе для всех видов доступа действуют права, задаваемые в Linux. Иными словами, если в `/data/dir1` есть файл, принадлежащий пользователю root, то пользователь peter, как правило, может только читать этот файл, но не может изменять. Такое же ограничение действует для всех пользователей сетевых каталогов.

Домашние каталоги

Если файловый сервер настроен для работы с большим количеством пользователей, то проще всего предоставить каждому пользователю Linux возможность видеть и изменять свой домашний каталог. Чтобы вам не пришлось заносить в `smb.conf` бесчисленное множество записей вида:

```
[username]
  User      = username
  writeable = yes
```

в smb.conf предусмотрена следующая сокращенная форма записи:

```
# в /etc/samba/smb.conf
...
[homes]
  writeable  = yes
  browseable = no
```

Таким образом, домашний каталог пользователя, активного в настоящий момент, виден под именем этого пользователя. Если применить параметр browseable=no, то пользователь не перестанет видеть свой домашний каталог, как это может показаться. Просто при этом каталог не будет отображаться в системе дважды: один раз под именем пользователя (например, peter), а один раз — под homes.

Групповые каталоги

Пользовательские и домашние каталоги позволяют пользователю сохранять свои файлы прямо на сервере, но при этом отсутствует возможность обмена данными, ведь эти каталоги невидимы для других пользователей, а значит, и недоступны. Для обмена данными применяются групповые каталоги, которыми могут пользоваться все члены определенной группы. Отнесение пользователей к группам происходит в рамках управления группами в Linux. Группа задается ключевым словом user и записью @*имя_группы*.

```
# в /etc/samba/smb.conf
...
[salesdata]
  user           = @sales
  path           = /data/sales
  writeable      = yes
  force group    = +sales
  create mask    = 0660
  directory mask = 0770
```

При настройке доступа к групповым каталогам особенно важно правильно указать, кто имеет право пользоваться какими ресурсами. Это же касается новых файлов и каталогов, которые будут создаваться впоследствии. Благодаря force group = +sales новые файлы и каталоги будут попадать в группу sales (а не в стандартную группу создающего их пользователя). Если пользователь не является членом группы sales, он не получит доступа к этому каталогу.

ВНИМАНИЕ

В вышеуказанном случае никогда не применяйте настройку force group = sales (то есть без плюсика), иначе любой акт доступа к каталогу будет осуществляться в Samba так, как будто активный в данный момент пользователь является членом группы sales — даже тогда, когда с точки зрения Linux пользователь совсем не входит в эту группу.

Иными словами, параметр force group = sales означает, что пользователи, не относящиеся к группе sales, получают право чтения и изменения файлов каталога. В групповых каталогах такая ситуация практически всегда неожиданна и может очень негативно повлиять на безопасность системы!

Параметры `create mask` и `directory mask` гарантируют, что члены группы смогут читать и изменять все новые файлы и каталоги, создаваемые другими членами той же группы (восьмеричное число соответствует значению `chmod` — см. `man chmod`). Если новые файлы или каталоги должны предоставляться членам группы только для чтения, но не для изменения, используйте значения `0440` и `0550`.

Каталоги, находящиеся в свободном доступе

Еще более «либеральным» является доступ к открытому для всех каталогу: каждый пользователь, который может пройти аутентификацию в Samba, может читать файлы из такого каталога. Доступ для изменения в данном случае отключается:

```
# в /etc/samba/smb.conf
...
[share]
  path      = /data/share
  read only = yes
```

Разумеется, вы можете настроить для каталогов со свободным доступом возможность внесения изменений (`writable = yes`). По умолчанию все пользователи имеют право читать файлы, созданные другими пользователями, но не могут их изменять. Для изменения этой ситуации используйте настройку `force group` и два параметра `mask`. Чтобы задействовать неограниченные взаимные права чтения и изменения файлов в группе, укажите `create mask=0666` и `directory mask=0777`.

Доступ для пользователей, не прошедших аутентификацию

Во всех предыдущих примерах предполагается, что пользователь может пройти аутентификацию в Samba. Однако Samba допускает и такую конфигурацию, при которой к каталогам получают доступ пользователи, не прошедшие аутентификацию. Они именуются *гостями* (guest). За работу с гостями отвечают глобальные настройки, обобщенные в следующем листинге:

```
# в /etc/samba/smb.conf
[global]
   ...
   map to guest       = bad user
   guest account      = nobody
```

Благодаря настройке `map to guest = bad user` попытки входа в систему под несуществующим логином автоматически присваиваются виртуальному пользователю Samba guest. По умолчанию в системе нет никаких сетевых каталогов или других ресурсов, которые может использовать guest. Параметр `guest account` указывает, каким пользователям Linux соответствуют гости. В большинстве дистрибутивов, в том числе в Debian и Ubuntu, для этого предусмотрен пользователь nobody.

Каталоги, которые должны быть открыты для использования гостями, обозначаются с помощью параметра `guest ok = ok`. Как правило, такие каталоги следует защищать от изменений — для этого применяется настройка `read only = yes`. Можно сказать, что гости смогут читать или изменять те же каталоги, что и пользователь Linux nobody.

```
[guest]
   path     = /data/guest
   guest ok  = yes
   read only = yes
```

Вариантом, аналогичным `guest ok` является команда `guest only = yes`: при такой настройке с каталогом могут работать не только гости, но и пользователи, прошедшие аутентификацию. Если вы вообще не хотите допускать гостей к использованию ресурсов Samba, задействуйте в разделе `[global]` настройку `map to guest = never`.

Совместные пользовательские каталоги (User Shares)

В версии 3.0.23 и выше Samba позволяет обычным пользователям, не обладающим правами администратора, самостоятельно предоставлять в совместное использование каталоги (User Shares). Соответствующие конфигурационные файлы обычно сохраняются в каталоге `/var/lib/samba/usershares` (по одному файлу на каждый предоставляемый каталог). В следующих строках показан пример такой операции:

```
#ВЕРСИЯ 2
# Файл /var/lib/samba/usershares/testdir
path     = /myhome/kofler/testdir
comment  =
guest_ok  = n
```

Детали конфигурации совместных пользовательских каталогов настраиваются в глобальном разделе `smb.conf` различными командами `usershare`. Команда `usershare allow` позволяет гостям предоставлять каталоги User Shares в свободное использование с `guest`-аккаунта (то есть без защиты паролем). Для этого при определении каталога нужно указать `guest ok` или `guest only`. Команда `usershare max shares` лимитирует количество совместных пользовательских каталогов.

```
# в /etc/samba/smb.conf
[global]
   ...
   usershare allow guests = yes
   usershare max shares = 100
```

Корзина для каталогов Samba

При удалении файлов в сетевом каталоге эти файлы, как правило, безвозвратно теряются. Функция «перемещения в корзину», предоставляемая в Linux, Windows или Mac OS X, действует только для файлов с локальной машины. Чтобы избежать

случайной потери файлов, можно создать подобную корзину и на уровне Samba. При простейшей конфигурации для этого достаточно добавить всего одну строку в файле `smb.conf`:

```
[global]
   vfs objects = recycle
```

Так активируется дополнительный модуль Samba `recycle` (VFS в данном случае означает Virtual File System, «виртуальная файловая система»). Теперь удаленные файлы по умолчанию будут оказываться в каталоге `.recycle` (расположенном относительно совместно используемого каталога). Но этот каталог, так или иначе, не отображается в большинстве файловых менеджеров. Чтобы сделать данный процесс более прозрачным, можно задать с помощью `recycle:repository` имя другого каталога, который будет использоваться в качестве корзины, например, вот так:

```
   recycle:repository = Trash
```

Окончательное уничтожение файлов происходит только тогда, когда они удаляются из корзины. В определенных обстоятельствах бывает целесообразно выполнять на сервере Samba специальный сценарий, который по истечении определенного срока автоматически удаляет старые файлы из корзины. Но чтобы этот механизм работал, потребуется обновить дату изменения при перемещении корзины (параметр `recycle:touch = Yes`). Различные другие параметры `recycle:xxx` документированы на сайте http://linux.die.net/man/8/vfs_recycle.

Параметры `recycle:xxx` можно устанавливать по вашему выбору — глобально или только для отдельных каталогов. Подобный пример приведен на сайте http://www.redhat.com/advice/tips/sambatrash.html.

Предоставление каталогов в общий доступ с помощью Gnome и KDE

Если вы желаете быстро и без лишних сложностей открыть каталог локальному пользователю через Samba, вам, конечно же, не захочется вручную вносить изменения в `smb.conf`. В таком случае каталог можно предоставить в совместное использование через специальное окно Gnome или KDE.

«За кулисами» как Nautilus (Gnome), так и Dolphin (KDE) применяют механизм Samba, связанный с предоставлением совместно используемых каталогов. Параметры каждого предоставляемого таким образом каталога сохраняются в отдельном конфигурационном файле в каталоге `/var/lib/samba/usershares`. Права доступа для этого каталога настраиваются так, что все пользователи, относящиеся к определенной группе (в Ubuntu это `sambashare`) могут создавать в этих каталогах новые файлы.

ПРИМЕЧАНИЕ

В некоторых дистрибутивах сначала требуется установить пакет nautilus-share*, предоставляющий в Nautilus диалоговые окна для обеспечения совместного использования каталогов. В некоторых обстоятельствах приходится самостоятельно установить и сконфигурировать Samba, прежде чем механизмы совместного использования начнут работать.

Nautilus/Gnome. В Nautilus щелкните на каталоге правой кнопкой мыши. Команда Параметры совместного использования из контекстного меню открывает простое диалоговое окно для конфигурации.

Dolphin/KDE. В KDE найдите в файловом менеджере Dolphin или Konqueror перейдите на вкладку в окне со свойствами Совместное использование. На данной вкладке можно создать сетевой каталог. Для этого в системе должна быть установлена как Samba, так и пакет `kdenetwork-filesharing`.

Управление паролями. Остается также открытым вопрос, кто сможет использовать предоставленный каталог. Если установить флажок Разрешить доступ гостю, то любой посетитель получит доступ к вашему сетевому каталогу. Если же снять этот флажок, то пользователь получит доступ к каталогу, только если войдет в систему под своим именем и паролем. Этот механизм работает только с пользователями, для которых на данном компьютере существует учетная запись, и лишь в том случае, если для них с помощью `smbpasswd` или `libpam-smbpass` был задан пароль Samba. Данные условия не отслеживаются ни в KDE, ни в Gnome. При необходимости нужно задать ваш пароль Samba с помощью `smbpasswd` в окне терминала.

20.5. Пример: домашний сервер/сервер мультимедиа

В этом разделе приведен простой пример конфигурации центрального сервера Samba. Представим себе семью, на трех компьютерах которой, используемых родителями и двумя детьми, накапливается все больше данных: цифровые фото, MP3, школьные работы, бухгалтерия и т. д. При децентрализованном хранении данных возникают некоторые проблемы.

- Не выполняется регулярное резервное копирование. (Что делать, если ноутбук потеряется по дороге в школу или сломается?)
- К общим данным обращаться достаточно сложно. Например, запланировано подарить бабушке на следующий день рождения альбом лучших семейных фотографий за последние годы. Но цифровые фото разбросаны по трем компьютерам и лежат в полном беспорядке. Похожие проблемы возникают и на вечеринках, когда вдруг оказывается, что музыка, которую все хотят услышать, находится на другом компьютере.
- Обмен данными между компьютерами усложнен и происходит в основном через флешки.

Наконец, сын, увлекающийся Linux, предлагает решить эти проблемы, оборудовав центральный домашний сервер или сервер мультимедиа. Сервер можно интегрировать в домашнюю сеть по беспроводной сети WLAN. При необходимости этот компьютер можно одновременно использовать как интернет-роутер и как брандмауэр.

В данном случае нас заинтересует только конфигурация с применением Samba: каждый член семьи получит собственный сетевой каталог, в котором он может считывать и изменять любые данные. Сохраняемые в этих каталогах файлы являются частными (с той оговоркой, что сын — системный администратор — может

прочитать и изменить любой файл). Для общего обмена данными создается еще пять каталогов. Родители могут обращаться к parents, дети — к children, а все члены семьи вместе — к family, audio и photos. Разумеется, каталоги audio и photos можно сделать просто подкаталогами family, но если выделить для них отдельные сетевые каталоги, то работать будет немного удобнее. При необходимости можно создать дополнительные учетные записи пользователей и каталоги.

Создание пользователей и групп Linux. В качестве имен пользователей мы будем применять слова mother, father, daughter, son. На практике, конечно же, будут использоваться настоящие имена, но здесь я от этого отказался, чтобы вам не пришлось запоминать имена членов выдуманной семьи. В командах useradd благодаря параметру --create-home создается каталог /home/*имя*. Затем в него копируется содержимое каталога /etc/skel. Если вам не нужны эти файлы, то можете сразу их удалить (последняя команда):

```
root# groupadd parents
root# groupadd children
root# groupadd family
root# useradd --create-home --groups parents,family father
root# useradd --create-home --groups parents,family mother
root# useradd --create-home --groups children,family son
root# useradd --create-home --groups children,family daughter
root# rm -rf /home/{father,mother,daughter/son}/*     (Необязательно)
```

Поскольку команда useradd выполнялась без пароля, новые пользователи были автоматически заблокированы (то есть они не могут войти в систему). Так и планировалось: мы не собирались давать членам семьи возможность входить на сервер под своими именами, так как в этом нет никакой необходимости.

Вместе с каждым пользователем создается и одноименная группа, являющаяся для него стандартной. Кроме того, новые пользователи попадают в группы family и parents или children. Таким образом, отец будет относиться к группам father, parents и family, мать — к группам mother, parents и family и т. д. Если позже вы захотите отнести пользователя еще к одной группе, то лучше всего применить следующую команду:

```
root# usermod -a -G newgroup user
```

Создание учетной записи пользователя Samba. Теперь создадим учетные записи пользователей Samba, чтобы каждая такая запись имела пароль:

```
root# smbpasswd -a father
New SMB password: *******
Retype new SMB password: *******
root# smbpasswd -a mother
...
root# smbpasswd -a son
root# smbpasswd -a daughter
```

Создание каталогов. При создании каталогов для общих файлов важно правильно настроить владельцев файлов и права доступа к файлам и каталогам, иначе в дальнейшем доступ к файлам может не функционировать. Благодаря первой команде chmod 770 только члены группы могут читать каталог и вносить в него

изменения. Вторая команда закрывает доступ к домашнему каталогу для любых пользователей, кроме его владельца.

```
root# mkdir /shared-data
root# mkdir /shared-data/{parents,children,family,audio,photos}
root# cd /shared-data
root# chown :parents parents/
root# chown :children children/
root# chown :famile family/ audio/ photos/
root# chmod 770 *
root# chmod 770 /home/{father,mother,son,daughter}
```

Важное достоинство общего файлового сервера заключается в том, что он облегчает централизованное резервное копирование. При этом защитить нужно только каталоги /home и shared-data.

Конфигурация Samba. В следующем примере показан конфигурационный файл smb.conf. Синхронизация паролей и любой доступ гостей к Samba деактивируются. Настройки различных каталогов должны быть понятны на основе материала, изложенного в разделе 20.4.

```
# /etc/samba/smb.conf для домашнего сервера
[global]
  workgroup          = home
  server string      = %h server (Samba, Ubuntu)
  security           = user
  passdb backend     = tdbsam
  unix password sync = no
  invalid users      = root
  map to guest       = never
  log file           = /var/log/samba/log.%m
  max log size       = 1000
  syslog             = 0
  dns proxy          = no
  panic action       = /usr/share/samba/panic-action %d
[homes]
  browseable         = no
  writeable          = yes
[parents]
  user               = @parents
  path               = /shared-data/parents
  writeable          = yes
  force group        = +parents
  create mask        = 0660
  directory mask     = 0770
[children]
  user               = @children
  path               = /shared-data/children
  writeable          = yes
  force group        = +children
  create mask        = 0660
  directory mask     = 0770
```

```
[family]
  user              = @family
  path              = /shared-data/family
  writeable         = yes
  force group       = +family
  create mask       = 0660
  directory mask    = 0770
[photos]
  user              = @family
  path              = /shared-data/photos
  ... как в [family]
[audio]
  user              = @family
  path              = /shared-data/audio
  ... как в [family]
```

В этой конфигурации есть один недостаток: все пользователи видят *все* каталоги, находящиеся в открытом доступе, в том числе те, которые предоставлены не для всех пользователей и которые могут применяться не всеми (например, родители видят каталог children, а дети — каталог parents). На самом деле пользоваться такими каталогами из-за настроек прав доступа удастся не всем, но красивее было бы, если бы эти каталоги вообще не были видны.

Такая возможность в конфигурации Samba, к сожалению, не предусмотрена. Некоторые каталоги можно скрыть с помощью настройки `browseable = no`, но тогда каталоги не сможет увидеть никто, даже их законные пользователи (каталоги при этом можно применять, как и раньше, но нужно точно указать путь вручную). Параметры `hide unreadable = yes` и `hide unwriteable = yes` в этом случае не помогут: С их помощью можно скрыть *внутри* сетевого каталога все файлы, которые нельзя читать и изменять определенному пользователю. Но сам сетевой каталог останется видим.

20.6. Пример: корпоративный сервер

Представим себе среднее предприятие, которое занимается производством измерительных приборов. в первую очередь, для обеспечения централизованного резервного копирования, но также и для упрощения обмена данными на фирме создается сервер Samba. Каждый сотрудник получает там собственный каталог. Кроме того, есть общие каталоги, перечисленные в табл. 20.1. В табл. 20.2 показано, к каким группам относятся сотрудники, в зависимости от должностей, занимаемых в компании.

Таблица 20.1. Обмен данными на фирменном сервере

Сетевой **каталог**	**Доступ (группы)**
Стратегия	Руководство
Бухгалтерия	Бухгалтерия, руководство
Разработка	Разработка, руководство
Продажи	Продажи, бухгалтерия, руководство

Таблица 20.2. Групповая принадлежность сотрудника в зависимости от занимаемой должности

Сотрудник	Принадлежность к группам
Руководство предприятия	Руководство, разработка, продажи, бухгалтерия
Бухгалтерия/учет	Бухгалтерия, продажи
Разработка	Разработка
Отдел продаж, маркетинг	Продажи

СОВЕТ

Если вы хотите сами выстроить схему распределения по группам или каталогам, то наивысшим приоритетом должна быть простота такой структуры. Чем больше существует групп и каталогов, тем более путаной будет система, тем неудобнее будет с ней работать и тем сложнее ее поддерживать. Лучше меньше, да лучше!

Даже если для создания Samba-сервера применяется дорогостоящее оборудование и гигабитные сетевые соединения, доступ к сетевым каталогам все равно будет протекать медленнее, чем к локальным файлам. Поэтому некоторые пользователи для ускорения работы будут пересохранять большие файлы на локальных машинах и работать с ними там. В таком случае обязательно следует применять автоматический сценарий, который будет ежедневно синхронизировать все локальные файлы с файлами из персонального сетевого каталога конкретного пользователя. Если каждый пользователь начнет сохранять свои файлы у себя на жестком диске, теряются все достоинства централизованного резервного копирования!

Создание пользователей и групп в Linux. Ради простоты, в качестве имен пользователей я буду далее применять в качестве имен пользователей `chief`*n*, `developer`*n* и т. д. На практике, разумеется, вы будете использовать реальные имена. Кроме имен, все команды очень похожи на примеры из предыдущего раздела (домашний сервер). Непосредственный вход сотрудника на сервер не предусмотрен, поэтому в данном случае нет этапа введения пароля.

```
root# groupadd management
root# groupadd sales
root# groupadd accountancy
root# groupadd developing
root# useradd --create-home \
      --groups management,sales,accountancy,developing chief1
root# useradd --create-home --groups sales sales1
root# useradd --create-home --groups sales sales2
root# useradd --create-home --groups developing developer1
root# useradd --create-home --groups developing developer2
root# useradd --create-home --groups accountancy,sales accountant1
root# useradd --create-home --groups accountancy,sales supervisor1
root# ...
```

Создание пользователя Samba. Далее создаем пользовательские записи, каждая запись — с паролем:

```
root# smbpasswd -a chief1
root# smbpasswd -a sales1
...
```

Создание каталогов. При создании каталогов для общих файлов важно правильно настроить владельцев файлов и права доступа — иначе доступ к файлам работать не будет. Благодаря первой команде `chmod 770` только члены конкретной группы могут читать и изменять каталог, а пользователи не могут получать доступ к чужим домашним каталогам.

```
root# mkdir /companydata
root# mkdir /companydata/{developing,sales,accountancy,strategy}
root# cd /companydata
root# chown :developing developing/
root# chown :sales    sales/
root# chown :accountancy accountancy/
root# chown :management     strategy/
root# chmod 770 /companydata/* /home/*
```

Конфигурация Samba. За исключением строки `workgroup`, глобальный раздел файла `smb.conf` выглядит точно так же, как и в примере с домашним сервером (см. раздел 20.5). Определение сетевого каталога также очень похоже на пример из предыдущего раздела. Разница заключается в том, что к каталогу могут обращаться члены сразу нескольких групп. Чтобы обмен данными работал без проволочек, очень важно задействовать `force group`. Все пользователи, имеющие право доступа к каталогу, должны относиться к группе, которой предоставляется такое право. Не забудьте поставить в файле `smb.conf` знак + перед именем группы!

```
# /etc/samba/smb.conf для корпоративного сервера (автономная конфигурация)
[global]
   ... Как и в предыдущем примере (домашний сервер)
[homes]
   browseable      = no
   writeable       = yes
[strategy]
   user            = @management
   path            = /companydata/strategy
   writeable       = yes
   force group     = +management
   create mask     = 0660
   directory mask  = 0770
[accountancy]
   user            = @accountancy, @management
   path            = /companydata/accountancy
   writeable       = yes
   force group     = +accountancy
   create mask     = 0660
   directory mask  = 0770
[developing]
   user            = @developing, @management
   path            = /companydata/developing
   writeable       = yes
   force group     = +developing
   create mask     = 0660
   directory mask  = 0770
```

```
[sales]
   user           = @sales, @accountancy, @management
   path           = /companydata/sales
   writeable      = yes
   force group    = +sales
   create mask    = 0660
   directory mask = 0770
```

20.7. Клиентский доступ

В этом разделе рассмотрим, как клиентский ПК будет обращаться к каталогам, которые Samba предоставила в совместное использование. Необходимо, чтобы выполнялось важное предварительное условие: порты TCP 135, 139 и 445, а также порты UDP 137 и 138 не должны блокироваться брандмауэром.

Клиенты Linux

Прежде чем ваши клиенты, которые работают с Linux, получат доступ к каталогам Windows, предоставленным Samba, вам, возможно, потребуется установить клиентские инструменты Samba (по умолчанию они обычно установлены). В Ubuntu необходимые программы упакованы в `samba-common`, `smbclient` и `libsambaclient`, в других дистрибутивах эти пакеты имеют немного другие названия.

KDE, Gnome. Для доступа к сетевому каталогу лучше всего использовать файловый менеджер Gnome или KDE. В обеих программах имеется средство для просмотра сети, которое сначала показывает все доступные каталоги Windows. Чтобы попасть в нужный каталог, вам понадобится пару раз щелкнуть кнопкой мыши и, возможно, ввести свои учетные данные (имя пользователя и пароль).

Пока файловый менеджер еще не в состоянии самостоятельно находить сетевые каталоги. В таком случае вы должны точно указать местоположение каталога в адресной строке файлового менеджера. Действует следующий принцип записи: `smb://`*`имя_сервера/название_каталога`*.

LinNeighborhood и pyNeighborhood. Нелюбители KDE и Gnome, которые все же ищут графическую систему для доступа к каталогам Windows, могут попробовать программу LinNeighborhood или ее новый вариант `pyNeighborhood`.

CIFS

Другой метод доступа к каталогам заключается в том, чтобы подключать сетевые каталоги непосредственно в локальное дерево каталогов, пользуясь при этом CIFS (общей межсетевой файловой системой). Разумеется, это нужно делать лишь в тех случаях, когда каталог доступен в течение достаточно долгого времени, то есть обслуживается стабильным сервером.

Система CIFS является «наследником» SMBFS (файловой системы Samba). CIFS-совместимые Samba-серверы передают UNIX/Linux-совместимую информацию о правах доступа к файлам, тогда как в SMBFS такая возможность отсутствует.

Работа с CIFS предполагает, что клиентские инструменты Samba и поддержка CIFS должны иметься в распоряжении `mount` (и, в частности, команды `mount.cifs`). В Debian и Ubuntu для этого требуется установить пакет `cifs-utils` (ранее `smbfs`). В Fedora и openSUSE этот пакет по умолчанию установлен.

Чтобы подключить к системе внешний каталог, введите одну из двух следующих команд (в зависимости от того, предоставляются ли каталоги Windows на базе имени пользователя):

```
root# mount -t cifs //jupiter/myshare /media/winshare
root# mount -t cifs -o username=имя //jupiter/myshare /media/winshare
```

Таким образом, каталог myshare подключается к файловой системе Linux на компьютере jupiter. Теперь его данные предоставлены в каталоге `/media/winshare`. Этот каталог должен существовать уже перед выполнением команды `mount`. При выполнении команд система запросит у вас пароль. Но пароль можно указать и сразу:

```
root# mount -t cifs -o username=имя,password=xxxxxxxx \
      //jupiter/myshare /media/winshare
```

Другие параметры mount. Чтобы можно было считывать и записывать файлы из папки пользователя в качестве обычного пользователя, укажите при вводе команды `mount` свои пользовательский номер и номер группы, который вы можете выяснить с помощью команды `id`.

```
root# mount -t cifs -o username= имя,password=xxxxxxxx,uid=1000,gid=1000 \
   //jupiter/myshare /media/winsha
```

Если имена файлов содержат международные символы и поэтому неверно отображаются, нужно добавить к команде `mount` параметр `iocharset=utf8`.

Хочу дать один совет по поводу NAS (Network Attached Storage, устройств хранения данных, подключаемых к сети). В некоторых версиях NAS применяются устаревшие версии Samba, функции DFS в которых реализованы с ошибками. (DFS означает Distributed Filesystem, «распределенная файловая система»; она позволяет единообразно обращаться к сетевым каталогам нескольких серверов). Даже если вы совсем не нуждаетесь в DFS, эти ошибки вызывают проблемы при взаимодействии с командой `rsync` (характерный признак — сообщения об ошибке `failed to set times on filename`). В таком случае используйте дополнительный `mount`-параметр `nodfs`.

/etc/fstab. Чтобы сетевой каталог всегда автоматически подключался к дереву каталогов, добавьте в `/etc/fstab` соответствующую запись, например, таким образом:

```
# в /etc/fstab
//jupiter/myshare /media/winshare cifs username=u,password=p,... 0 0
```

В большинстве дистрибутивов все CIFS-каталоги, перечисленные в `/etc/fstab`, подключаются к системе во время Init-процесса. Исключением является SUSE: здесь за этот процесс отвечает сценарий Init-V `cifs` (в старых версиях openSUSE — `smbfs`), который требуется явно активизировать:

```
root# insserv cifs
```

Выгрузка учетных данных. По соображениям безопасности не рекомендуется непосредственно указывать пароль в /etc/fstab. Лучше выгрузить эту информацию в отдельный файл, который может читать только администратор (root). Кроме того, создайте файл /etc/.winshare-pw, в котором будет содержаться пароль и (если потребуется) рабочая группа (домен) для сетевого каталога. Файл имеет следующий вид:

```
username=name
password=xxxx
domain=workgroup
```

Следующая команда ограничивает доступ к файлу. Теперь только администратор сможет читать и изменять этот файл:

```
root# chmod 600 /etc/.winshare-pw
```

Добавьте также к записи fstab параметр credentials. При подключении каталога файлы аутентификации считываются из .winshare-pw.

```
# Дополнение в /etc/fstab
//jupiter/myshare  /media/winshare cifs credentials=/etc/.winshare-pw 0 0
```

Команды smbclient и smbtree

Smbclient. Любители текстовых команд также могут просматривать сетевые каталоги с помощью команды smbclient. Она пусть и не очень удобна, но часто помогает отследить источник проблемы, возникающей в Samba.

Команда smbclient -L localhost отображает все ресурсы локального компьютера, предоставленные в общее пользование, перечисляет все видимые рабочие группы локального компьютера и указывает, какой компьютер в какой группе является основным (ведущим). Если ресурс не защищен паролем, то в ответ на запрос пароля просто нажимайте клавишу Enter. В случае если на локальном компьютере не работает сервер Samba, то вместо localhost укажите имя этого компьютера.

Если smbclient выдает сообщение о невозможности входа (access denied), это обычно означает, что имя пользователя или рабочей группы на вашем компьютере Linux не совпадает с аналогичным именем на компьютере Windows или на сервере Samba. Простейшее решение — сообщить эту информацию smbclient как дополнительные параметры:

```
user$ smbclient -U имя_пользователя -W рабочая_группа -L jupiter
```

Вы можете также применять smbclient интерактивно, для передачи файлов. Для этого сначала нужно установить соединение между открываемым каталогом и компьютером Windows или сервером Samba. Каталог указывается в привычной для Windows манере — *имя_сервера**название_каталога*. Чтобы оболочка не пыталась обработать символы \, их нужно удваивать. Затем, как и при использовании FTP, вы можете просматривать каталоги командой ls, командой cd переходить в другой каталог, с помощью get переносить данные на локальный компьютер (скачивать), а с помощью put сохранять данные на внешнем компьютере (закачивать). Список

важнейших команд выводится с помощью help. Подробное описание команд дается в man smbclient.

```
user$ smbclient -U name -W wgname \\\\jupiter\\myshare
Password: xxxxxx
Domain=[wgname] OS=[Unix] Server=[Samba 3.5.4]
smb: > ls
 .           D      0  Thu Sep  7 17:38:02 2010
 ..          D      0  Thu Sep  7 17:38:02 2010
 data        D      0  Wed Apr  5 18:17:11 2010
 file.xy     AR   226  Sat Dec 14 00:00:00 2010
```

Smbtree. Команда smbtree выводит древовидный список всех серверов Samba и Windows, которые находятся в сети, включая все объекты, предоставляемые на этих серверах для совместного использования. Как правило, smbtree использует актуальное имя пользователя и запрашивает соответствующий пароль. С помощью -user=name%password можно настроить эти данные при вызове команды. Чтобы найти ресурсы, доступные без пароля, используйте параметр -N. В следующем листинге показаны два компьютера (kofler-desktop и ubuntu-test), одна виртуальная машина (merkurvm), а также один подключаемый к сети жесткий диск (wd-nas), все эти элементы находятся в рабочей группе WORKGROUP.

```
root# smbtree
Enter kofler's password: ********
WORKGROUP
    \\KOFLER-DESKTOP            kofler-desktop server (Samba, Ubuntu)
        \\KOFLER-DESKTOP\mydata
        \\KOFLER-DESKTOP\IPC$ IPC Service (kofler-desktop server)
        \\KOFLER-DESKTOP\print$  Printer Drivers
    \\MERKURVM                  merkurvm
        \\MERKURVM\images
        \\MERKURVM\data
        \\MERKURVM\SharedDocs
    \\UBUNTU-TEST               ubuntu-test server (Samba, Ubuntu)
        \\UBUNTU-TEST\IPC$    PC Service (ubuntu-test server ...)
        \\UBUNTU-TEST\print$     Printer Drivers
    \\WD-NAS                 My Book World Edition Network Storage
        \\WD-NAS\IPC$         IPC Service (My Book ...)
        \\WD-NAS\ConfigurationSystem Configuration
        \\WD-NAS\multimedia
        \\WD-NAS\Download     Download Share
        \\WD-NAS\Public       Public Share
```

Клиенты Windows

В старых версиях Windows все серверы Samba отображались непосредственно при просмотре сети и ресурсов в Проводнике или в окне для выбора файлов. Часто заново установленный и сконфигурированный сервер Samba становится виден в сети только через несколько минут. Надежнее в таком случае просто перезапустить компьютер с Windows.

LLTD. К сожалению, Windows Vista, Windows 7 и другие современные версии этой системы не распознают в локальной сети ни сервер Samba, ни старые серверы Windows. Виноват в этом новый протокол, применяемый для обмена сетевыми данными, а именно Link Layer Topology Discovery (LLTD, обнаружение топологии уровня связи). Этот протокол очень неплох, так как работает значительно быстрее своих предшественников. К сожалению, он пока не поддерживается ни Samba, ни более ранними версиями Windows. Более подробная информация о LLTD содержится на сайтах http://support.microsoft.com/kb/922120 и http://www.microsoft.com/whdc/rally/rallyLltd.mspx.

К счастью, доступ к сетевым устройствам, не совместимым с LLTD, возможен и без их автоматического распознавания. Просто задайте в Проводнике Windows имя компьютера в форме `\\имя` вручную.

Разумеется, более красиво было бы обеспечить совместимость сервера Linux с LLTD. Как ни странно, Microsoft разработала программу специально для этой цели и бесплатно предоставила ее исходный код. К сожалению, данный код не отвечает лицензии свободного ПО, поэтому эта программа по умолчанию не поставляется ни с одним из дистрибутивов Linux.

Кроме того, в модуле Net::Frame содержится свободная реализация на Perl.

21 NFS и AFP

В предыдущей главе мы поговорили о протоколе SMBP (блок серверных сообщений), подробно рассмотрев программу Samba, построенную на его основе. Samba уже стала настоящим «общим знаменателем» при обмене данными между различными операционными системами. Но есть и альтернативные подходы и инструменты. В этой главе мы поговорим о классическом протоколе UNIX NFS (Сетевая файловая система), а также о протоколе AFP (Apple Filing Protocol), предпочитаемом в OS X.

21.1. NFS 4

Сетевая файловая система (NFS) позволяет предоставлять компьютерам, находящимся в сети, доступ к локальным каталогам других компьютеров, также расположенных в сети. В отличие от SMB (Samba), в данном случае сетевой каталог подключается к клиентскому компьютеру командой `mount` либо соответствующей строкой в `/etc/fstab`, оказываясь прямо в файловой системе. Следовательно, вы не можете просто взять и выбрать сетевой каталог в файловом менеджере, совершив всего пару щелчков кнопкой мыши. NFS рассчитана на работу с такой конфигурацией, которая изменяется редко и при которой клиентские компьютеры требуют постоянного доступа к NFS-серверу.

Базовые функции NFS входят в состав ядра, таким образом достигается максимальная скорость работы. В качестве альтернативы вы можете воспользоваться NFS-сервером для пользовательского пространства, но он уже практически не применяется, и в этой главе мы не будем его рассматривать.

Интегрированные в ядро функции NFS поддерживают NFS-версии 3 и 4. Версия NFS 4 уже считается вполне доработанной, и ей следует отдавать предпочтение. Только в редких случаях, например, когда приходится работать с очень старыми клиентскими компьютерами, не поддерживающими NFS 4, может потребоваться прибегнуть к NFS 3.

Серверная конфигурация

Ядро Linux уже поддерживает NFS 4 по умолчанию. Но все же для работы с NFS необходимо установить пакеты `nfs-common` и `nfs-kernel-server` (Debian, Ubuntu)

или nfs-utils (Fedora, RHEL, SUSE). Содержащиеся в них программы и сценарии отвечают за запуск необходимых сетевых служб.

Наиболее важное предварительное условие для работы с сервером NFS 4 заключается в том, что должна функционировать служба rpc.idmapd. Эта программа обеспечивает соотнесение (ассоциирование) имен пользователей в NFS и идентификаторов UID/GID. Порядок запуска этой службы отличается в зависимости от дистрибутива.

- В Debian за запуск rpc.idmapd отвечает сценарий Init-V nfs-common. Он запускает службу автоматически, если существует файл /etc/exports (на сервере) или если в /etc/fstab содержится строка nfs4 (на клиенте). При необходимости возможен и принудительный запуск. Для этого нужно использовать в /etc/default/nfs-common настройку NEED_IDMAPD=yes.
- В RHEL, а также в сравнительно старых версиях Fedora rpc.imapd по умолчанию запускается с помощью /etc/init.d/nfs. Специальная конфигурация не требуется. В Fedora 16 и выше требуется выполнить две следующие команды:

```
root# systemctl start nfs-idmap.service
root# systemctl enable nfs-idmap.service
```

- В SUSE необходимо убедиться, что в /etc/sysconfig/nfs содержится настройка NFS4_SUPPORT=yes (по умолчанию это так). После этого rpc.idmapd запускается системой Init.
- В Ubuntu за запуск idmapd отвечает файл /etc/init/idmapd.conf.

Подготовка сетевых каталогов. Все каталоги, которые предоставляются в совместное использование («экспортируются») с помощью NFS, должны быть подкаталогами корневого каталога, который служит псевдофайловой системой. Это проще всего объяснить на примере. Предположим, вы хотите экспортировать уже имеющиеся каталоги /data/audio, /data/fotos и /iso-images. Для этого нужно создать четыре новых каталога, причем /nfsexport выступает в качестве корневого каталога (имя этого каталога является произвольным). Каталоги /nfsexport/audio, /data/fotos и /iso пусты. Они служат лишь в качестве точек подключения (mount point).

```
root# mkdir /nfsexport
root# mkdir /nfsexport/audio
root# mkdir /nfsexport/fotos
root# mkdir /nfsexport/iso
```

Теперь подключаем /data/ и /data/fotos как новые подкаталоги nfsexports. Таким образом, содержимое каталога /data/audio теперь будет видимым и в /nfsexport/audio, а содержимое /data/fotos — в /nfsexport/fotos.

```
root# mount -t none -o bind /nfsexport/audio/   /data/audio/
root# mount -t none -o bind /nfsexport/fotos/   /data/fotos/
root# mount -t none -o bind /nfsexport/iso/     /iso-images/
```

Чтобы впоследствии подключение NFS-каталога к системе происходило автоматически, добавьте в файл /etc/fstab (на сервере) следующие строки:

```
# /etc/fstab
...
```

```
/data/audio     /nfsexport/audio none bind 0 0
/data/fotos     /nfsexport/fotos none bind 0 0
/iso-images     /nfsexport/iso none bind 0 0
```

Файл /etc/exports. Файл /etc/exports — это основной конфигурационный файл NFS. Он определяет, какой компьютер к каким файлам будет иметь доступ и каким именно будет этот доступ. Компьютеры можно указывать либо по IP-адресам, либо по сетевым именам. IP-адреса можно маскировать (например, с помощью 192.168.0.0/255.255.255.0 или 192.168.0.0/24). В названиях компьютеров могут содержаться джокерные символы * (например, *.sol), а в IP-адресах — нет.

Учтите, что между IP-адресами или хост-именами и, соответственно, параметрами, не должно быть никаких пробелов! Если вы хотите предоставить каталог для совместного использования без ограничений на всех компьютерах, способных установить соединение с NFS-сервером, просто укажите символ *.

В следующем примере изображено, что все клиенты с IP-адресами в сети 192.168.0.* или с именем *.lan могут обращаться к каталогам /nfsexport/audio и /nfsexport/fotos. Кроме того, все компьютеры, находящиеся в сети (без ограничений по IP-адресу или хост-имени) могут обращаться к /nfsexport/iso. Каталоги audio и iso открыты только для чтения. Длинные определения экспорта разбиваются на две строки с помощью символа \.

```
# Файл /etc/exports fur NFS 4
/nfsexport          192.168.0.0/24(rw,async,no_subtree_check,fsid=0,crossmnt) \
                    *.lan(rw,async,no_subtree_check,fsid=0,crossmnt)
/nfsexport/audio    192.168.0.0/24(ro,async,no_subtree_check) \
                    *.lan(ro,async,no_subtree_check)
/nfsexport/fotos    192.168.0.0/24(rw,async,no_subtree_check) \
                    *.lan(rw,async,no_subtree_check)
/nfsexport/iso      *(ro,async,no_subtree_check)
```

Синтаксис /etc/exports понятен из предыдущих строк. За названием каталога и хост-именем или IP-адресом в скобках следуют различные параметры NFS, наиболее важные из которых кратко перечислены ниже (несколько других параметров описаны в man exports).

- ro (read-only) или rw (read-write) — указывают, предоставляется ли доступ только для чтения или для чтения и внесения изменений.
- sync или async — определяют момент, в который NFS-сервер подтверждает изменения, внесенные в файл. По умолчанию действует sync. При такой настройке информация сохраняется лишь тогда, когда файл действительно сохранен. Параметр async гораздо эффективнее, но вместе с тем и ненадежнее. При доступе для внесения изменений скорость работы sync и async отличается радикально (до 10 раз), поэтому на практике async применяется довольно часто.
- no_subtree_check или subtree_check — указывают, должен ли NFS-сервер тестировать поддерево. Коротко расскажу об этом процессе: если по NFS экспортируется каталог файловой системы (но не вся файловая система), то NFS-сервер тестирует поддерево, определяя, таким образом, находится ли в экспортируемом каталоге конкретный файл. Затем NFS-сервер передает клиенту информацию о фактическом местоположении файла. Если позже файл на сервере будет

переименован, то на клиентском компьютере возникнут проблемы. По этой причине по умолчанию в современных версиях NFS-сервера тестирование поддерева отключено. И все же параметр no_subtree_check нужно указать, чтобы не получать при запуске системы соответствующие предупреждения от сервера.

При необходимости можете активизировать тестирование поддерева. В man exports это рекомендуется делать, в первую очередь, с каталогами, файлы в которых переименовываются редко и экспортируются в режиме «только для чтения».

- Пользователь root, конечно, может использовать NFS, как и любой другой пользователь, но в экспортированных каталогах из соображений, связанных с безопасностью, он имеет только права пользователя nobody (UID=65534 и GID=65534). Если вы хотите дать root его обычные права, укажите в /etc/exports параметр no_root_squash.
- Корневой каталог NFS 4 помечается параметром fsid=0. Этот параметр может использоваться только с корневым каталогом! (В NFS 4 отсутствует возможность экспорта каталогов, находящихся вне корневого каталога!)
- Параметр crossmnt также может сопровождать лишь корневой каталог. Благодаря этому параметру при подключении подкаталогов их содержимое остается видимым на клиентских машинах и в том случае, когда корневой каталог на конкретном клиенте не подключен. Вместо того, чтобы задавать параметр crossmnt на корневом каталоге, также можно указать параметр nohide для всех подкаталогов — эффект будет аналогичным.

Exportfs -a. Если NFS-сервер уже работает, то после внесения любых изменений в /etc/exports нужно выполнить команду exportfs -a. Так мы гарантируем, что NFS-сервер будет учитывать измененные новые записи.

```
root# exportfs -a
```

Настройки, характерные для отдельных дистрибутивов. Кроме стандартных конфигурационных файлов, вы можете указать и такие настройки, которые являются специфичными для отдельных дистрибутивов:

- Debian, Ubuntu — /etc/defaults/nfs-common, /etc/defaults/nfs-kernel-server;
- Fedora, SUSE, Red Hat — /etc/sysconfig/nfs.

Запуск. В Debian и Ubuntu сервер NFS по умолчанию запускается сразу же после установки. В Fedora, Red Hat и SUSE вам придется помочь компьютеру, как и при запуске других служб Init-V, выполнив следующие команды:

```
root# systemctl start nfs-server.service (Fedora)
root# systemctl enable nfs-server.service
root# chkconfig --level 35 nfs on        (RHEL)
root# service nfs start
root# insserv nfs                        (SUSE)
root# service nfs start
```

UID и GID-ассоциирование. Для ассоциирования имен пользователей и наименований групп в NFS 4 обычно применяется ID-ассоциирование. Файл, который принадлежит пользователю hofer и находится на NFS-сервере, доступен пользова-

телю hofer и на клиентском компьютере с NFS для чтения и изменения. Группы и отдельные пользователи, присутствующие на сервере и отсутствующие на клиенте, относятся на сервере к пользователю и группе nobody.

В принципе ID-ассоциирование в NFS 4 работает гораздо интеллектуальнее, чем в NFS 3, где вся система ассоциирования построена на UID- и GID-номерах. И все же ассоциирование на NFS 4 имеет свои сюрпризы: необходимо обязательно убедиться, что пользователи применяют на всех компьютерах сети одни и те же имена аккаунтов!

UID- и GID-ассоциирование обеспечивает уже упоминавшийся выше демон rpc.idmapd. Его конфигурация осуществляется в файле /etc/idmapd.conf. Для минимальной конфигурации NFS 4, описываемой в этой главе, можно оставить файл в таком виде, в каком он по умолчанию предоставляется в вашем дистрибутиве.

Брандмауэр. В NFS 4 вся коммуникация осуществляется через TCP-порт 2049. Если ваш сервер защищен брандмауэром, то этот порт нужно предоставить в совместное использование в локальной сети.

NFS 4 с LDAP и Kerberos. В этом пункте мы поговорим о конфигурации сервера без системы аутентификации, то есть о простейшем способе использования NFS 4. В больших сетях со многими пользователями вам, как правило, потребуется соединить NFS 4 с LDAP и Kerberos. Эта конфигурация довольно сложна, и ради экономии места я не буду ее здесь описывать. Соответствующие объяснения можно найти в Интернете, например на следующих сайтах:

- http://www.itp.uzh.ch/~dpotter/howto/kerberos;
- http://wiki.debian.org/nfs4-kerberos-ldap;
- http://www.danbishop.org/2012/06/02/ubuntu-12-04-ultimate-server-guide-first-draft/.

Клиентская конфигурация

Чтобы NFS 4 можно было использовать на клиентском компьютере, нужно установить соответствующий пакет, зависящий от дистрибутива. Это nfs-common (Debian, Ubuntu), nfs-utils (Fedora, RHEL) или (SUSE), Кроме того, на компьютере должен работать демон rcp.idmapd.

Mount и unmount. Чтобы можно было применять сетевой каталог, его нужно интегрировать в дерево каталогов командой mount. Следующие команды интегрируют в локальную файловую систему все дерево каталогов nfsexport. Это происходит в точке /media/nfsdata. При этом имя jupiter нужно заменить хост-именем NFS-сервера. Обратите внимание: корневой каталог NFS адресуется просто символом /, а не /nfsexport!

```
root# mkdir /media/nfsdata
root# mount -t nfs4 jupiter:/ /media/nfsdata
root# ls /media/nfsdata
audio fotos iso
```

В качестве альтернативы, можно импортировать лишь часть каталогов:

```
root# mkdir /media/fotos
root# mount -t nfs4 jupiter:/fotos /media/fotos
```

Команда umount снова удаляет каталог NFS из локальной файловой системы. Если сетевое соединение уже прервано, umount нужно выполнять с параметром -f. Или же придется долго ждать, пока umount выполнится!

```
root# umount /media/fotos
root# umount /media/nfsdata
```

СОВЕТ

В openSUSE для создания NFS-каталогов лучше использовать YaST-модуль Сетевые службы ▸ NFS-клиент. Когда я пытался вручную подключить к openSUSE 12.2 NFS-каталоги, у меня ничего не получилось, так как не запускался демон rpc.idmapd (service start nfs и insserv nfs выдавали сообщения об ошибках. Но с YaST все работало вполне хорошо).

/etc/fstab. Чтобы при запуске компьютера автоматически интегрировать NFS-каталоги в файловую систему, дополните файл /etc/fstab строкой, построенной по следующему образцу. В четвертом столбце можно использовать NFS-специфичный параметр bg. В таком случае mount попытается в фоновом режиме подключить сетевой каталог, если вы не предоставите его сразу. Это удобно, в первую очередь, в тех случаях, когда сетевые каталоги требуется автоматически подключать при запуске компьютера.

```
# Дополнение в /etc/fstab
jupiter:/fotos     /media/fotos nfs4 bg 0 0
```

Automount/ autofs. Если в большой сети есть много NFS-каталогов, редко бывает удобно активизировать их все с помощью /etc/fstab. Это стоит времени и ресурсов, даже в случае, если большинство из этих каталогов почти не будет использоваться. Гораздо удобнее уже при первом применении автоматически подключать новые каталоги к дереву каталогов. Во многих дистрибутивах эту задачу решает пакет autofs или autofs4.

Поиск ошибок

Если mount-команду на клиенте выполнить не удается, попытайтесь проверить, не произошло ли следующих ошибок.

- Соединение не должно блокироваться брандмауэром. При работе с NFS 4 должен быть свободен TCP-порт 2049.
- Демон rpc.idmapd должен работать как на сервере, так и на клиенте. Чтобы убедиться, что так и есть, выполните ps ax | grep idmapd.
- На сервере должна работать программа NFS-сервер. Чтобы в этом убедиться, выполните rpcinfo -p. Из приведенных ниже строк следует, что на сервере функционирует NFS для версий 2, 3 и 4.

```
root# rpcinfo -p | grep nfs
  program vers proto port  service
  100003    2    tcp  2049  nfs
  100003    3    tcp  2049  nfs
  100003    4    tcp  2049  nfs
  ...
```

- С помощью `showmount -e` проверьте, какие каталоги предоставлены для совместной работы с NFS. Пусть вас при этом не смущают такие указания, как * или `everyone` — `showmount` не в состоянии интерпретировать более подробную информацию о правах доступа:

```
user@nfsserver$  showmount -e
/nfsexport       (everyone)
/nfsexport/audio (everyone)
/nfsexport/fotos (everyone)
/nfsexport/iso   *
```

  `showmount` также можно выполнить и на клиентском компьютере, но при этом нужно указать хост-имя NFS-сервера:

```
user$client#    showmount -e <nfsserver>
```

- Убедитесь в том, что клиент и сервер расположены в одном и том же сегменте сети.

21.2. NFS 3

Хотя NFS 4 существует уже не первый год, система NFS 3 по-прежнему широко распространена. Не в последнюю очередь это объясняется исключительно простой конфигурацией. Но у NFS 3 есть недостатки.

- Ассоциирование владельцев файлов и групп происходит на базе ID-номеров. Файл на сервере, принадлежащий пользователю с UID 1000, имеет такой же UID-номер и на клиенте — даже если аккаунты с UID 1000 на сервере и на клиенте не совпадают. Таким образом для работы с NFS 3 необходимо, чтобы на всех компьютерах локальной сети применялись одни и те же UID- и GID-номера!
- Защита системы с помощью брандмауэра организуется гораздо сложнее. NFS использует протоколы TCP и UDP на портах 111 (`portmap`) и 2049 (`nfsd`), а также на случайных свободных портах (`rpc.*d`).
- Функции блокировки и подключения к дереву каталогов не интегрированы непосредственно в NFS 3, а должны выполняться специальными программами (`portmap` и `rpc.mountd`).
- NFS 3 не отвечает за кодировку имен файлов. Если на сервере и на клиенте применяются разные кодировки, то имена файлов отображаются неправильно. (Но в Linux это, как правило, не представляет проблемы, так как во всех распространенных дистрибутивах уже много лет применяется кодировка Unicode UTF8.)
- NFS 3 не поддерживает ни списки контроля доступа (ACL), ни аутентификацию с применением Kerberos или SPKM 3.

Отличное руководство по NFS 3 с большим количеством советов по настройке и обеспечению безопасности приводится на сайте http://nfs.sourceforge.net/nfs-howto/. Но учтите, что о NFS 4 там ничего не сказано.

Серверная конфигурация

/etc/exports. Настроить сервер для работы с NFS 3 проще, чем с NFS 4. Не требуется сначала собирать все предназначенные для экспорта каталоги в собственном каталоге с помощью `mount`. Вместо этого можно без дополнительной подготовки перечислять эти каталоги прямо в `/etc/exports`. За исключением этого синтаксис NFS 3 практически такой же, как в NFS 4.

Различия касаются лишь параметров экспорта.

- Параметры `fsid` и `crossmnt` в NFS 3 не действуют.
- Для экспорта должен применяться параметр `insecure`, если NFS-сервер должен быть доступен и для компьютеров Apple с системой OS X. Благодаря этому параметру сервер NFS реагирует и на такие клиентские запросы, которые приходят с IP-порта с номером более 1024. Именно так работают NFS-клиенты с OS X.

В следующем примере с `exports` экспортируются те же три каталога, что и в примере с NFS 4:

```
# Файл  /etc/exports для NFS 3
/data/audio 192.168.0.0/24(ro,async,no_subtree_check) \
                *.lan(ro,async,no_subtree_check)
/data/fotos 192.168.0.0/24(rw,async,no_subtree_check) \
                *.lan(rw,async,no_subtree_check)
/data/iso-images    *(ro,async,no_subtree_check)
```

Exportfs -a. Как и в случае с NFS 4, после проведения изменений в файле `exports` системы NFS 3 нужно выполнить команду `exportfs -a`:

```
root# exportfs -a
```

UID и GID. Для управления правами доступа к файлам и каталогам в NFS 3 применяются UID и GID. Такая система проста, но работает лишь при условии, что на сервере и на всех клиентах соблюдается однозначное соответствие между пользователями, группами и их ID-номерами.

При управлении пользователями вручную достичь однозначного соответствия между UID и GID обычно бывает очень сложно и для этого требуется особая тщательность: создавая на любом компьютере новую пользовательскую учетную запись, нужно вручную задавать UID и GID. Кроме того, вам понадобится центральный справочный список всех уже розданных номеров UID и GID. Если при управлении пользователями вы допустите ошибки или любую небрежность, то сразу же столкнетесь с неприятными последствиями: например, если пользователи peter@merkur и birgit@neptun, работающие на разных компьютерах, имеют одинаковый UID 1234, то у обоих будут одинаковые права при доступе к каталогу NFS. А такого лучше не допускать.

Раньше подобная проблема решалась путем синхронизации файлов `/etc/passwd`, `/etc/group` и `/etc/shadow` на всех компьютерах локальной сети с помощью NIS (сетевой информационной службы). Настроить NIS достаточно просто, но она считается устаревшей и ненадежной. Гораздо лучше применять централизованное управление пользовательскими данными (логин, пароль, групповая отнесенность, UID и GID и т. д.) с помощью службы LDAP. LDAP означает Lightweight Directory

Access Protocol (Облегченный протокол доступа к каталогу) — это протокол для управления иерархическими данными. В качестве LDAP-сервера в Linux обычно применяется программа openLDAP. К сожалению, конфигурировать и эксплуатировать сервер LDAP достаточно сложно.

Брандмауэр. Чтобы закрыть доступ к серверу NFS 3, нужно просто заблокировать порты 111 и 2049 для протоколов TCP и UDP. Но, к сожалению, обратное невозможно: нельзя открыть полностью заблокированную систему только для NFS 3. Дело в том, что NFS 3 использует наряду с портами 111 и 2049 и случайные порты, которые в настоящий момент свободны. Чтобы справиться с этой проблемой, можно жестко соотнести порты со службами `statd`, `lockd` и `mountd`. В RHEL, например, это делается в файле `/etc/sysconfig/nfs`:

- http://www.gotdoug.com/?p=3;
- http://docs.redhat.com/docs/en-US/Red_Hat_Enterprise_Linux/6/html/Storage_Administration_Guide/s2-nfs-nfs-firewall-config.html.

Файлы /etc/hosts.allow, /etc/hosts.deny. В файлах `hosts.allow` и `hosts.deny` указывается, какие компьютеры имеют право обращаться к NFS-серверу. Информация, содержащаяся в `/etc/exports`, важна только для пользователей, которые в принципе могут связаться с NFS-сервером. В этом отношении файлы `hosts.allow` и `hosts.deny` занимают первое место в иерархии средств защиты доступа. Синтаксис этих файлов описан в разделе 26.2. Для NFS-сервера релевантны записи `portmap` и `mountd`.

Обратите внимание, что данные `/etc/hosts.allow` и `-.deny` учитываются для серверов NFS 3, но не для серверов NFS 4!

Mount-статус. Команда `showmount -a` позволяет вывести список всех активных NFS-клиентов. Обратите внимание, что эта команда действует только в NFS 3! Каталоги, предоставляемые сервером в совместное использование с помощью NFS 4, в выводе этой команды не отображаются.

Клиентская конфигурация

Клиентская конфигурация в NFS 3 минимально отличается от подобной конфигурации в NFS 4. Основное предварительное условие заключается в том, что необходимо установить на клиенте специальные пакеты, которые отличаются от дистрибутива к дистрибутиву. Это `nfs-common` (Debian, Ubuntu), `nfs-utils` (Fedora, RHEL) или `nfsclient` (SUSE). При выполнении команды `mount` вручную вы указываете в качестве типа файловой системы `nfs`, а не `nfs4`. Кроме того, потребуется полностью задать путь к сетевому каталогу на NFS-сервере:

```
root# mount -t nfs jupiter:/data/fotos /media/fotos
```

Аналогичные правила действуют и для записей в файле `/etc/fstab`:

```
# Дополнение в /etc/fstab
jupiter:/data/fotos      /media/fotos nfs bg 0 0
```

22 SSH (Secure Shell)

Если у вас нет физического доступа к тому или иному серверу, то администрировать такой сервер вы можете только через сетевое соединение. Лучше всего для этого подходит инструмент SSH (Secure Shell). Он обеспечивает как простое выполнение команд, так и работу с программами, имеющими графический интерфейс. При этом устанавливается надежное сетевое соединение. Единственное предварительное условие заключается в том, что на сервере должна быть инсталлирована специальная программа SSH-сервер. В этой главе даются советы о том, как можно максимально быстро и надежно установить SSH-сервер.

SSH — потомок протоколов telnet и rlogin, и он значительно надежнее их обоих. С клиентской точки зрения SSH уже был рассмотрен в разделе 10.2. Более подробно о клиентском и о серверном применении этого инструмента рассказано на сайте OpenSSH: http://www.openssh.com/.

Корневой сервер. Эта глава о SSH — первая из нескольких, в которых будут описаны сетевые службы, обычно работающие на корневом сервере. *Корневой сервер* — это внешний сервер, находящийся в вычислительном центре, причем вы можете самостоятельно администрировать такой сервер без каких-либо ограничений (имея права администратора). Для опытного системного администратора Linux корневой сервер — удобная и недорогая возможность создавать собственные веб-ресурсы, использовать свой почтовый сервер и т. д. Такая возможность интересна, в первую очередь, для небольших и средних предприятий, подключающихся к Интернету по технологии ADSL. Подобные компании не приспособлены для использования серверных служб под собственным доменным именем.

При выборе корневого сервера необходимо, в первую очередь, проверять, насколько серьезна компания, предоставляющая хостинг. Нет ничего более неприятного, чем неработающая система поддержки компьютера, к которому невозможен физический доступ. Важна и административная помощь — в частности, веб-интерфейс, позволяющий после аварийного завершения сеанса перезагрузить сервер либо запустить «живую» систему — и таким образом исправить дефектную конфигурацию.

Наконец, учитывайте, какие дистрибутивы Linux можно установить на сервере — не каждый провайдер поддерживает все распространенные дистрибутивы. Наряду с (дорогими) корпоративными дистрибутивами для применения на корневом сервере хорошо подходят Debian, Ubuntu, LTS и CentOS. Поскольку техни-

ческая поддержка конкретных версий Fedira и openSUSE длится сравнительно недолго, использовать их в данном качестве не рекомендуется.

22.1. Установка

Во многих дистрибутивах программа для SSH-сервера попадает на жесткий диск уже при первой инсталляции дистрибутива. Только если SSH-сервер не устанавливается автоматически, вам придется решить эту проблему самостоятельно.

Debian, Ubuntu. В Debian и Ubuntu нужно установить пакет `openssh-server`. Демон `sshd` сразу же запустится автоматически.

```
root# apt-get install openssh-server
```

Fedora, RHEL. В Fedora и RHEL установка происходит аналогичным образом, с помощью `yum`.

```
root# yum install openssh-server
```

В Fedora SSH-сервер сразу же выполняется автоматически. В RHEL, напротив, приходится вручную осуществлять как первый запуск программы, так и ее активизацию.

```
root# service sshd start (Первый автоматический запуск SSH-сервера,
                          только в RHEL)
root# service --add sshd (Последующие автоматические запуски SSH-сервера,
                          только в RHEL)
```

OpenSUSE. В openSUSE интересующий нас пакет называется `openssh`. В современных дистрибутивах openSUSE с поддержкой Systemd `openssh` сразу же запускается автоматически.

```
root# zypper install openssh
```

22.2. Конфигурация и защита

Конфигурационные файлы `sshd` находятся в каталоге `/etc/ssh`. За серверную конфигурацию отвечает `sshd_config`. Как правило, этот файл можно оставить без изменений, так как SSH-сервер должен заработать «с ходу». Обмен информацией по умолчанию осуществляется через IP-порт 22. Если у вас есть вопросы, на страницах `man` вы найдете много дополнительной информации.

Безопасный FTP. Составной частью SSH-сервера является SFTP-сервер. Это безопасный аналог обычного FTP-сервера. Обычно функции SFTP предоставляются автоматически, пока работает SSH-сервер. Правда, работать с ними могут только SFTP-совместимые клиенты (например, программа `sftp`). В SFTP предусмотрены лишь пользовательские учетные записи (то есть отсутствует анонимный FTP).

Обеспечение безопасности. В принципе SSH-сервер начинает работать без дополнительной конфигурации. Но в этом кроется значительный риск, который

нельзя недооценивать: любой, кто угадает рабочую комбинацию логина и пароля, может войти в систему с вашего компьютера! Взломщики применяют автоматические инструменты, которые ищут серверы в Интернете и пытаются на них зайти. Все подобные действия регистрируются в файле `/var/log/auth.log`. На общедоступных серверах случается по тысяче попыток взлома в день! По этой причине вам очень не помешает позаботиться о том, чтобы все пользователи применяли как можно более непростые пароли. Например, можно использовать команду `makepasswd` из одноименного пакета, которая создает надежные пароли.

В файле `/etc/shadow` в зашифрованном виде сохранены все пользовательские пароли, и в нем ни в коем случае не должно быть записей без паролей! Эти записи выделяются тем, что в строке между первым и вторым двоеточиями отсутствует текст. Обычно там должен находиться или зашифрованный пароль, или — в системных учетных записях — специальный символ (* или !). Эти символы полностью исключают возможность входа в систему. Если в файле действительно найдется запись без пароля, исправьте эту ошибку, указав *пароль имя*.

Соблюдая перечисленные ниже правила, вы значительно снизите вероятность взлома вашего SSH-сервера. Описанные методы можно применять вместе или по отдельности.

Отсутствие учетной записи администратора. Агрессор пытается заполучить административные права, а проще всего это сделать, войдя в систему с учетной записи администратора. Для этого требуется угадать всего один параметр — пароль администратора. Гораздо безопаснее будет запретить вход в систему от имени администратора, если он осуществляется по SSH. Иначе говоря, по SSH нужно войти в систему под другой учетной записью, а потом перейти на учетную запись администратора с помощью команд `su` или `sudo` (проверьте, функционирует ли этот механизм, прежде чем вносить следующее изменение).

```
# Изменение в /etc/ssh/sshd_config
...
PermitRootLogin = no
```

Чтобы данное изменение вступило в силу, `sshd` должна заново считать конфигурационные файлы:

```
root# service ssh reload
```

В результате такого приема агрессору теперь будут неизвестны два параметра: *и* логин, *и* пароль!

Альтернатива `PermitRootLogin = no` — это настройка `without-password`:

```
# Изменение в /etc/ssh/sshd_config
...
PermitRootLogin = without-password
```

Не пугайтесь, такая настройка ни в коем случае не допускает входа с правами администратора без указания пароля! Команда `without-password` означает, что вход в систему будет возможен и в дальнейшем, но для аутентификации потребуется ввести пароль. Обычно при этом происходит обмен ключами (подробнее о такой технологии рассказано ниже).

Изменение SSH-порта. По умолчанию обмен информацией сервером происходит через порт 22. В строке `port` вы можете без труда указать другой порт, который в настоящее время не используется. Поскольку многие автоматические инструменты взлома отслеживают только порт 22, вы одновременно избавляетесь от массы проблем, связанных с безопасностью.

Если вы работаете с командой `ssh`, то при каждом ее применении нужно специально указывать порт для SSH-сервера с параметром `-p`. Обратите внимание, что при работе с `scp` нужно применять параметр `-P`, так как `-p` в этой программе имеет значение `preserve` и этот параметр сохраняет данные о времени доступа к копируемой информации и другие данные о доступе.

Разумеется, не забывайте и о том, что защита методом изменения порта доступа не является абсолютной. Если ваш сервер серьезно атакован, а не просто кто-то ищет любой плохо защищенный сервер, чтобы установить на нем свой вирус, то атакующие обязательно просканируют все порты. Тогда рано или поздно враг обнаружит ваш SSH-сервер, независимо от того, на каком порту он работает.

Кроме того, метод изменения порта имеет еще один недостаток, которым нельзя пренебречь. Конфигурация большинства брандмауэров настроена так, что они не перекрывают трафик, идущий через порт 22, а остальные порты вполне могут перекрываться. Если вы, например, работаете на некотором предприятии всего пару дней и хотите быстро войти на сервер через SSH, то брандмауэр фирменной сети вас скорее всего не пропустит.

TCP-Wrapper. SSH-сервер использует особую библиотеку, которая называется TCP-Wrapper. Тем не менее вы вполне можете определять с помощью конфигурационных файлов `/etc/hosts.allow` и `/etc/hosts.deny`, какие сетевые адреса могут использовать SSH-сервер. Когда на корневом сервере функционирует SSH-сервер, защита такого рода редко бывает целесообразной — ведь SSH должен быть доступен для работы с адресами со всего Интернета. Если же вы, напротив, установили SSH-сервер на сервере локальной сети и администрирование должно осуществляться только в рамках этой сети, будет очень разумно ограничить соответствующим образом и права доступа. Подробно о библиотеке TCP-Wrapper и о соответствующих конфигурационных файлах мы поговорим в разделе 26.2.

DenyHosts. Сценарий DenyHosts, написанный на языке Python, отслеживает все попытки входа в систему по SSH. Если он устанавливает, что с определенного IP-адреса было предпринято много неудачных попыток входа, то этот IP-адрес автоматически добавляется в файл `/etc/hosts.deny`. После этого в зависимости от конфигурации данный IP-адрес либо навсегда остается в `hosts.deny`, либо удаляется оттуда по истечении определенного времени (например, через день или через неделю).

DenyHosts достаточно эффективно предотвращает попытки автоматического входа в систему (они часто имеют место при атаке, которая осуществляется сразу с нескольких различных компьютеров). На серверах, с которыми доводилось работать мне, эта программа действовала очень хорошо. Во многих дистрибутивах она содержится в готовых пакетах. Подробнее о наиболее актуальной версии этого сценария рассказано на сайте http://denyhosts.sourceforge.net/.

DenyHosts запускается сценарием Init-V и интерпретирует конфигурационный файл `/etc/denyhosts.conf`. Необходимо, чтобы программа `denyhost` отслеживала

нужный файл логов (параметр SECURE_LOG). В следующем листинге показано несколько строк конфигурации моего веб-сервера http://kofler.info:

```
# Файл /etc/denyhosts.conf
SECURE_LOG = /var/log/auth.log
HOSTS_DENY = /etc/hosts.deny

# Через 24 часа вновь высвобождаем IP-адреса, которые были заблокированы
PURGE_DENY = 1d
# Блокировка после трех неуспешных попыток входа под неверным логином
DENY_THRESHOLD_INVALID = 3

# Блокировка после пяти неуспешных попыток входа под верным логином
DENY_THRESHOLD_VALID = 5
# Блокировка после неуспешной попытки входа для root
DENY_THRESHOLD_ROOT = 1
```

22.3. Аутентификация с помощью ключей

При работе с SSH-сервером лучше всего использовать для аутентификации SSH-сервер. В таком случае аутентификация осуществляется не с применением пароля, а с использованием ключа. Для работы по такому принципу на локальном компьютере с помощью ssh-keygen создается пара ключей. Подобный ключ вы самостоятельно зашифровываете фразой-паролем. (Фраза-пароль — это пароль, состоящий из нескольких слов.) Кроме того, вы добавляете открытый ключ. Он также аутентифицируется паролем, а сама операция осуществляется командой ssh-copy-id. Открытый ключ записывается в файл .ssh/authorized_keys на сервере:

```
user@client$ ssh-keygen
Generating public/private rsa key pair.
Enter file in which to save the key (/home/user/.ssh/id_rsa): <Return>
Enter passphrase (empty for no passphrase): ********
Enter same passphrase again: ********
Your identification has been saved in /home/user/.ssh/id_rsa.
Your public key has been saved in /home/user/.ssh/id_rsa.pub.
user@client$ ssh-copy-id -i user@server
user@server's password: *******
```

Если задать ответ на запрос фразы-пароля, просто нажав Enter или введя empty, то ssh-keygen не производит шифрования. Это удобно, поскольку в таком случае вы сможете пользоваться SSH, не вводя пароль. Но возникает риск с точки зрения безопасности: если злоумышленнику в руки попадет ключ, находящийся на клиентском компьютере, то этот человек сможет без всяких проблем входить в систему с компьютеров, на которых вы установили открытую часть ключа!

Команда ssh-copy-id не срабатывает, если сервер допускает только аутентификацию по ключу. Для решения этой проблемы, напоминающей пресловутую проблему «курицы и яйца», нужно выполнить передачу ключа с другого клиентского компьютера, который может аутентифицироваться на сервере. В таком случае

необходимо передать на сервер файл открытого ключа ~/.ssh/id_rsa.pub через третий компьютер и там вручную вставить нужную информацию в конце файла .ssh/authorized_keys.

Теперь после установки соединения с целевым компьютером ssh выполнит обмен информации, относящейся к ключам. Вход с логином после этого не требуется, но нужно ввести фразу-пароль, которой зашифрована закрытая часть ключа.

Ssh-agent. Пока ваши ключи защищены паролем (в том числе фразой-паролем), вы можете работать с уверенностью, что информация находится в безопасности — но сами будете испытывать при работе определенные неудобства. Довольно безопасное и при этом удобное решение подобной проблемы предлагает ssh-agent. Эта программа управляет всеми закрытыми ключами пользователя (но обычно такой ключ только один). Программа запускается следующим образом:

```
user$ eval $(ssh-agent)
```

В результате изменяются некоторые переменные окружения консоли, используемой в настоящий момент. ssh-agent продолжает работать в качестве фонового процесса. С помощью ssh-add вы можете добавить ваши закрытые ключи:

```
user$ ssh-add ~/.ssh/id_rsa
Enter passphrase for /home/user/.ssh/id_rsa: ******
```

Начиная с этого момента, ssh использует файлы ключей, управляемые ssh-agent. И это означает, что система никогда больше не будет запрашивать у вас пароль к файлу ключей. Не придется вводить пароль для каждой ssh-команды, это потребуется сделать всего лишь однажды. Серьезный недостаток ssh-agent заключается в том, что область действия изменяемых переменных окружения ограничивается только одной консолью.

Gnome-keyring. Демон gnome-keyring-daemon по умолчанию занимается в Gnome работой с паролями и SSH-ключами. Для первоначального доступа к данным обычно требуется корректный вход в систему (но не автоматический вход через дисплейный менеджер!) либо ввод главного (master) пароля. Для администрирования сохраненных ключей и паролей применяется программа seahorse.

Проблемы с входом в систему. Вход без пароля на SSH-сервер не всегда гладко функционирует. Это объясняется различными настройками системы безопасности. Если SSH-сервер работает в режиме STRICT_MODE (по умолчанию — именно так), то файл authorized_keys учитывается лишь в том случае, когда достаточно жестко заданы права доступа. Если файл ключей создан программой ssh-copy-id, права задаются не так жестко, как требуется. В подобной ситуации помогают две следующие команды:

```
user@server$ chmod 700 ~/.ssh
user@server$ chmod 600 ~/.ssh/authorized_keys
```

На серверах Fedora и RHEL часто возникает и другая проблема: файл ключей, созданный с помощью ssh-copy-id, не содержит контекстной информации, необходимой для работы с SELinux. Вот как с этим справиться:

```
root@server# /sbin/restorecon -r /root/.ssh       (Для администратора)
root@server# /sbin/restorecon -r /home/user/.ssh  (Для остальных пользователей)
```

СОВЕТ

Если SSH-логин действует не так, как вы ожидали, выполните команду ssh- с дополнительным параметром -v. После этого ssh выдает множество отладочных сообщений. Данный параметр можно указывать до трех раз (-v -v -v). При этом соответственно повышается детализация протоколирования, выполняемого ssh.

Полный отказ от входа в систему с паролем. Когда у вас будет нормально работать создание SSH-соединения с ключом (то есть без необходимости входа в систему с логином), можно попробовать вообще деактивировать вход с паролем. Для этого измените на сервере конфигурационный файл `sshd_config`. Наиболее важны две строки:

```
# в /etc/ssh/sshd_config
...
PasswordAuthenticationno
UsePAM                  no
```

В таком случае SSH-аутентификация становится осуществимой *только* с ключами. Будьте внимательны, чтобы не выставить себя из собственной системы! Если вы потеряете ключ к своему клиентскому компьютеру, то не сможете войти на сервер! Так бывает всегда: чем выше уровень безопасности, тем меньше гибкость всей системы...

23 Apache

Apache — это веб-сервер номер один из мира свободного ПО. Согласно данным netcraft.com, по состоянию на май 2012 года около 57 % всех сайтов работают с Apache. Если же учитывать только миллион наиболее посещаемых сайтов, то доля Apache на рынке оказывается просто баснословной — 62 %.

В этой главе мы поговорим о версии Apache 2.2. Это может показаться удивительным — ведь более новая версия 2.4 доступна уже с начала 2012 года. Но все корпоративные дистрибутивы пока продолжают использовать версию Apache 2.2 и не спешат обновляться. Актуальная информация, а также подробная документация по Apache приводится на официальном сайте сервера: http://www.apache.org.

23.1. Установка и настройка Apache

Установка, запуск и тестирование

Установка. Обычно установленная копия Apache состоит из нескольких взаимосвязанных пакетов: самого сервера, различных библиотек, плагинов, языков программирования и др. Чтобы облегчить установку, в некоторых дистрибутивах предусмотрен выбор для инсталляции сразу группы пакетов. При этом вместе с пакетами Apache устанавливаются важнейшие пакеты для работы с PHP и MySQL.

```
root# yum groupinstall 'Web-Server'          (Fedora, RHEL)
root# zypper intall -t pattern lamp_server   (SUSE)
root# tasksel install lamp-server            (Ubuntu)
```

ПРИМЕЧАНИЕ

Начиная с версии 2 Apache поддерживает три различных режима многопоточной работы: perchild, prefork и worker. От выбранного метода зависит, насколько эффективно Apache сможет синхронно обрабатывать несколько запросов. При установке Apache вам необходимо выбрать один из трех этих вариантов. Если вы собираетесь использовать вместе с Apache язык программирования PHP, то лучше всего указать prefork. При работе с другими вариантами возможны ошибки, так как их библиотеки не приспособлены к применению потоков (http://www.php.net/manual/en/faq.installation.php).

Запуск/остановка. Apache — это демон, который в некоторых дистрибутивах нужно специально запускать. Важнейшие команды для управления им рассмотрены в разделе 8.5. Названия сценария Init-V в разных дистрибутивах могут быть различными: `apache2` в Debian, SUSE и Ubuntu или `httpd` в Fedora и Red Hat.

Брандмауэр. В Fedora, Red Hat и (open)SUSE брандмауэр, активный по умолчанию, блокирует доступ к веб-серверу извне. Поэтому сначала вы можете протестировать Apache лишь непосредственно на том компьютере, на котором работает веб-сервер (http://localhost). Чтобы веб-сервер также был доступен извне, вы должны определить в настройках брандмауэра исключения протоколов HTTP и HTTPS (то есть для портов 80 и 443).

Название программы и учетная запись. Названия программы, представляющей собой веб-сервер Apache, в разных дистрибутивах также различаются. По причинам, связанным с безопасностью, веб-сервер, подобно многим другим сетевым демонам, выполняется не от имени администратора, а под другой учетной записью. Чтобы узнать ее название, лучше всего выполнить `ps axu`. В табл. 23.1 показано, что названия программы и учетной записи в различных дистрибутивах неодинаковы.

Таблица 23.1. Название программы, учетная запись и каталог DocumentRoot Apache

Дистрибутив	Название программы	Учетная запись	DocumentRoot
Debian, Ubuntu	apache	www-data	/var/www
Fedora, Red Hat	httpd	apache	/var/www/html
SUSE	httpd2-thread*метод*	wwwrun	/srv/www/htdocs

Тестирование. Чтобы проверить, все ли работает, откройте браузер и введите в адресную строку `http://localhost/` или `http://`*`имя_сервера`*`/`. Вы должны увидеть тестовую страницу веб-сервера (рис. 23.1).

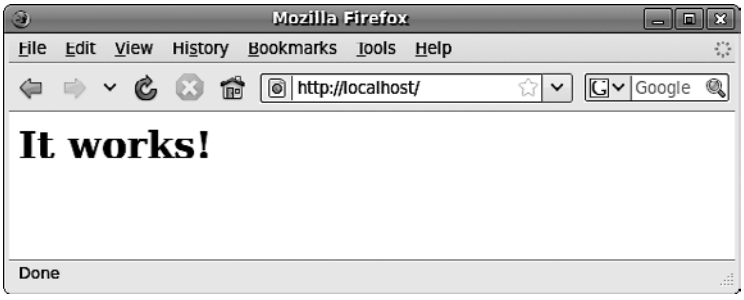


Рис. 23.1. Тестовая страница Apache на компьютере с Ubuntu

Собственные HTML-страницы. Чтобы вместо тестовой страницы на экране появилась главная страница вашего сайта, нужно сохранить HTML-файлы в каталоге документов Apache. Этот каталог также по-разному называется в различных дистрибутивах (ключевое слово `DocumentRoot` в конфигурационных файлах, см. табл. 23.1). Ваши HTML-файлы должны быть доступны для чтения с той учетной записи, под которой выполняется веб-сервер Apache!

Конфигурация

Размеры книги не позволяют подробно описать конфигурацию Apache. Но хотя бы рассмотрим, где располагаются конфигурационные файлы различных дистрибутивов и как производятся простейшие настройки.

Ранее конфигурация Apache выполнялась в файле `httpd.conf`, точное местоположение которого зависело от дистрибутива. Но со временем этот конфигурационный файл становился все более путаным. Вместе с этим возрастала сложность автоматизированной технической поддержки, в частности активации и деактивации плагинов.

В результате в большинстве дистрибутивов стал использоваться другой метод — настройки распределялись между несколькими файлами, а затем считывались из различных каталогов с помощью команд `include` (табл. 23.2–23.4). Такой метод позволяет сделать отдельные файлы более понятными, но сама система становится совсем запутанной. Кроме того, в таком случае практически невозможно перенести настройки одного дистрибутива в другой. Если вы ищете в конфигурационных файлах определенное ключевое слово, то лучше всего действовать так:

```
user$ cd /etc/httpd (или) cd /etc/apache2
user$ find -type f -exec grep -i -q ключевое_слово ^; -print
```

Таблица 23.2. Конфигурация Apache в Debian и Ubuntu

Файлы	Содержание
/etc/apache2/apache2.conf	Исходный пункт
/etc/apache2/httpd.conf	Пользовательская конфигурация
/etc/apache2/ports.conf	Отслеживаемые порты, обычно — 80
/etc/apache2/mods-available/*	Доступные дополнительные модули
/etc/apache2/mods-enabled/*	Ссылки на активные дополнительные модули
/etc/apache2/conf.d/*	Прочие конфигурационные файлы
/etc/apache2/sites-available/*	Доступные сайты (виртуальные хосты)
/etc/apache2/sites-enabled/*	Ссылки на активные сайты
/etc/apache2/envvars	Переменные окружения для сценария Init-V

Таблица 23.3. Конфигурация Apache в Fedora и Red Hat

Файлы	Содержание
/etc/httpd/conf/httpd.conf	Исходный пункт
/etc/httpd/conf/magic	Конфигурация типов MIME (для mod_mime)
/etc/httpd/conf.d/*.conf	Файлы для конфигурации модулей

Таблица 23.4. Конфигурация Apache в SUSE

Файлы	Содержание
/etc/apache2/httpd.conf	Исходный пункт
/etc/apache2/*.conf	Глобальные конфигурационные файлы
/etc/apache2/sysconf.d/*.conf	Автоматически генерируемые системные конфигурационные файлы
/etc/apache2/conf.d/*.conf	Прочие конфигурационные файлы
/etc/apache2/vhosts.d/*.conf	Сайты (виртуальные хосты)
/etc/sysconfig/apache2	Основные настройки

В Debian/Ubuntu в каталоге mods-available содержится коллекция файлов `*.load` и `*.conf` для различных модулей Apache. Для активизации других модулей создайте

в mods-enabled ссылки на эти файлы. При управлении ссылками в Debian вам пригодятся специфичные для этого дистрибутива команды a2enmod и a2dismod. В дальнейшем вы сможете активизировать и деактивировать виртуальные хосты с помощью команд a2ensite и a2dissite. По умолчанию в sites-available содержится только файл default: он конфигурирует стандартную веб-страницу сервера (каталог /var/www), а кроме того, содержит разнообразные базовые настройки для протоколирования ошибок и обращений к страницам.

Механизм работы такой же, как и с модулями: в каталоге sites-available содержатся все конфигурационные файлы для всех хостов, а в sites-enabled располагаются соответствующие ссылки.

В SUSE все CONF-файлы из каталога sysconf.d при каждом запуске Apache создаются сценарием Init-V /etc/init.d/apache2 заново! По этой причине вносить изменения в эти файлы бессмысленно. Напротив, вам следует изменить переменные, находящиеся в /etc/sysconfig/apache2. Кроме того, в данном файле определяется, какие модули должны загружаться при запуске Apache (переменная APACHE_MODULES). Если хотите добавить к конфигурационным файлам SUSE собственный файл, укажите его название в переменной APACHE_CONF_INCLUDE_FILES.

Тестирование конфигурации. Изменив синтаксис, с помощью команд httpd -t, httpd2 -t или apache2 -t вы можете проверить, нет ли в конфигурации синтаксических ошибок. В Debian и Ubuntu сначала нужно считать из файла envvars некоторые переменные окружения:

```
root# . /etc/apache2/envvars
root# apache2 -t
Syntax OK
```

После этого прикажите Apache заново считать конфигурационные файлы:

```
root# service apache2|httpd reload
```

Переменная ServerName. Обычно веб-сервер Apache запускается сразу. Но в зависимости от настроек конкретной сети, вам потребуется изменить или добавить в конфигурационные файлы как минимум одну строку: переменная ServerName должна содержать имя вашего компьютера. Если эта настройка не подействует, укажите UseCanonicalName Off.

```
# в   /etc/apache2/httpd.conf         (Debian/Ubuntu)
# или /etc/httpd/conf/httpd.conf      (Fedora/Red Hat)
ServerName mars.sol     # Здесь укажите имя вашего компьютера
```

В SUSE имя компьютера записывается в файле /etc/sysconfig/apache2 в переменной APACHE_SERVERNAME.

Стандартная кодировка

Во всех распространенных дистрибутивах Linux автоматически применяется кодировка Unicode UTF-8. Иначе говоря, если вы создаете в текстовом редакторе новый текстовый файл, в котором есть специальные символы, например немецкие буквы ä, ö, ü или ß, они сохраняются в кодировке UTF-8.

Для Apache, в принципе, неважно, в какой кодировке сохранены файлы. Программа просто переносит файлы байт за байтом в браузер, запросивший страницу. Apache посылает вместе со страницей так называемый *заголовок*, где в числе прочего указано, в какой кодировке создана страница. Браузер интерпретирует эту информацию и использует указанную кодировку при отображении сайта.

Настройка кодировки. Теперь Apache должен правильно указать кодировку. Если этого не получится сделать, пользователь увидит у себя в браузере не ä или ü, а какие-нибудь причудливые комбинации символов. Во избежание такого в Apache предусмотрена возможность настройки кодировки.

- `AddDefaultCharset off` — при таком значении Apache интерпретирует `<meta>`-тег передаваемого HTML-файла и сообщает браузеру, какая кодировка указана в этом теге. Если файл HTML начинается так, как это показано ниже, то применяется кодировка Unicode UTF-8:

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">
<html><head>
<meta http-equiv="Content-Type" content="text/html;charset=utf-8" />
...
```

- `AddDefaultCharset charset` — Apache сообщает указанную здесь кодировку всем страницам браузера. Настройка действует для HTML- и PHP-файлов. `<meta>`-тег в HTML-коде игнорируется.
- `AddDefaultCharset charset extension` — так настраивается кодировка для файлов, имеющих определенное расширение. Если указать `AddCharset utf-8 .utf8`, то все файлы, название которых оканчивается на `.utf8`, будут посланы в браузер в кодировке Unicode UTF-8. Для работы `AddCharset` требуется модуль Apache `mod_mime`.

Debian, Ubuntu. Разумеется, стандартная конфигурация в разных дистрибутивах неодинакова. Для глобальной настройки кодировки в Ubuntu предусмотрен конфигурационный файл `/etc/apache2/conf.d/charset`. По умолчанию он пуст, то есть действует `AddDefaultCharset off`.

Кроме того, `AddDefaultCharset` и `AddCharset` можно использовать в конфигурационных файлах для виртуальных хостов (каталог sites-available), а также в файлах `.htaccess`, если вам нужна особая конфигурация отдельно взятого хоста или каталога. Но не забывайте, что настройки кодировки, указываемые в `.htaccess`, учитываются лишь в тех случаях, когда для веб-каталога задан параметр `AllowOverride All` или `FileInfo`.

Fedora, Red Hat. В Fedora и Red Hat также может применяться `AddDefaultCharset UTF-8`. Эта настройка находится в файле `/etc/httpd/conf/httpd.conf`. В том же файле располагается настройка `AllowOverride None` для каталога `/var/www/html`.

SUSE. В SUSE отсутствуют особые функции для задания кодировки в конфигурационных файлах. Таким образом, действует `AddDefaultCharset off`, то есть кодировка определяется только по данным, указанным в `<meta>`-теге HTML-файла. Для настройки `AddDefaultCharset` хорошо подходит файл `/etc/apache2/mod_mime-defaults.conf`. Кроме того, в SUSE действует `AllowOverride None` для каталога `/srv/www/htdocs`. Эту настройку можно изменить в файле `/etc/apache2/default-server.conf`.

Logrotate

На многих серверах файлы логов Apache считаются одними из тех, которые разрастаются наиболее быстро. Поэтому необходимо регулярно переименовывать, архивировать и, например, удалять файлы логов. Именно такую задачу выполняет программа `logrotate` (см. раздел 11.7), которая, как правило, по умолчанию устанавливается на серверах Linux.

Эта программа обычно запускается раз в день программой `/etc/cron.daily/logrotate`. В стандартной конфигурации она обрабатывает файлы логов Apache `/var/log/httpd/*.log` (Fedora/RHEL) или `/var/log/apache2/*.log` (Debian/Ubuntu) раз в неделю, переименовывает их в *имя.nn* и архивирует. Заархивированные данные сохраняются в течение 52 недель, а затем удаляются.

Если при конфигурировании виртуальных хостов вы определяете собственные директории логов, то нужно соответствующим образом откорректировать в конфигурационном файле `/etc/logrotate.d/apache2` первую строку и указать там места, где будут храниться дополнительные файлы логов. При этом допускаются и такие варианты, как `/home/*/www-log/*.log`.

```
# Файл /etc/logrotate.d/apache2
/var/log/apache2/*.log /home/meinefirma/www-log/*.log {
   weekly
   missingok
   rotate 52
   ...
}
```

23.2. Создание и защита веб-каталогов

После завершения базовой конфигурации Apache вы, как правило, будете создавать различные веб-каталоги. В них будут находиться те HTML- и PHP-файлы, из которых состоит ваш сайт. На следующих страницах я исхожу из того, что используется стандартная конфигурация Apache, как, например, в Debian и Ubuntu. Если вы работаете с другими дистрибутивами, то в них стандартная конфигурация немного отличается от описанной. Но представленные здесь ключевые слова и способы работы действуют и там.

Стандартная конфигурация Ubuntu. В Ubuntu стандартная конфигурация Apache такова, что для стандартного веб-сайта используются файлы из каталога `/var/www`. Необходимые настройки находятся в файле `/etc/apache2/sites-available/default`. Приведенный далее файл, кроме того, содержит различные базовые настройки (логирование, активизация именных виртуальных хостов и т. д.).

```
# Файл /etc/apache2/sites-available/default (Ubuntu)
<VirtualHost *:80>
  ServerAdmin      webmaster@localhost
  DocumentRoot     /var/www/

 # Строгие ограничительные стандартные настройки для всех каталогов
 <Directory />
```

```
 Options          FollowSymLinks
 AllowOverride    None
</Directory>

# Предварительные настройки для каталога DocumentRoot
<Directory /var/www/>
 Options          Indexes FollowSymLinks MultiViews
 AllowOverride    None
 Order            Allow,Deny
 Allow            from all
</Directory>

# CGI-сценарии
ScriptAlias /cgi-bin/ /usr/lib/cgi-bin/
<Directory "/usr/lib/cgi-bin">
 AllowOverride    None
 Options          +ExecCGI -MultiViews +SymLinksIfOwnerMatch
 Order            Allow,Deny
 Allow            from all
</Directory>

# Файлы документации (доступ только с localhost)
Alias /doc/ "/usr/share/doc/"
<Directory "/usr/share/doc/">
 Options          Indexes MultiViews FollowSymLinks
 AllowOverride    None
 Order            Deny,Allow
 Deny             from all
 Allow            from 127.0.0.0/255.0.0.0 ::1/128
</Directory>

# Настройки логирования
LogLevel          warn
ErrorLog          ${APACHE_LOG_DIR}/error.log
CustomLog         ${APACHE_LOG_DIR}/access.log combined
<VirtualHost>
```

Особенность конфигурационного файла `default` заключается в том, что все его настройки объединены в группе `<VirtualHost>`. Группы `<VirtualHost>` применяются для того, чтобы отделять друг от друга многочисленные самостоятельные хосты (веб-сайты). Хост в файле `default` в любом случае не имеет связи ни с IP-адресом, ни с хост-именем и по этой причине действует для всех операций веб-доступа, которые не могут быть отнесены к специальному виртуальному хосту.

Конфигурационные строки из каталогов `/cgi-bin/` и `/doc/`, как правило, можно закомментировать (на случай, если вы захотите использовать CGI-сценарии).

Конфигурация хоста

Ключевые слова, которые описаны ниже, применяются при конфигурировании группы `<VirtualHost>` и содержат детальное описание хоста, то есть информацию

о происхождении данных, адрес электронной почты администратора, местонахождение файлов логирования и т. д.

- **DocumentRoot** — указывает, в каком каталоге находятся HTML-файлы.
- **ServerAdmin** — задает адрес электронной почты администратора виртуального хоста. Этот адрес отображается, например, при сообщениях об ошибках. Здесь нужно указать адрес электронной почты, который действительно активен. Обычно он имеет вид `webmaster@hostname`.
- **ServerSignatur** — определяет, должен ли Apache добавлять сигнатуру в конце самостоятельно сгенерированных сообщений (например, сообщений об ошибках, списках каталогов и т. д.). В сигнатуре указывается версия Apache, а также хостимя. С помощью `ServerSignatur=EMail` также можно задать адрес электронной почты администратора.
- **Loglevel** — определяет, в каком объеме должны протоколироваться проблемы, возникающие с веб-сервером. Диапазон значений простирается от `emerg` (протоколируются только критические ошибки, приводящие к завершению работы Apache) до `debug` (протоколируется вся информация, даже отладочные тексты). Как правило, целесообразно задавать настройки `error` или `warn`.
- **ErrorLog** — дает имя файла с протоколом, записывая это имя в сообщениях об ошибках.
- **CustomLog** — указывает имя файла протокола доступа. В этом файле Apache записывает все успешные операции переноса файлов. В качестве второго параметра вы передаете либо имя заранее определенного формата логирования, либо строку с собственными командами форматирования. Допустимые коды форматов описаны по адресу http://httpd.apache.org/docs/2.2/de/mod/mod_log_config.html.

  В Ubuntu в файле `apache2.conf` некоторые форматы заданы заранее, например `combined` или `common`.
- **ErrorDocument** — указывает, как Apache должен реагировать на ошибки. В качестве первого параметра задается номер ошибки (например, 404 для `not found` (не найдено)), а в качестве второго параметра — имя локального файла или адрес внешнего сайта, который в таком случае будет отображаться. Имя файла должно указываться относительно `DocumentRoot`. Важнейшие коды ошибок:
  - 400 — Bad Request (неверный запрос);
  - 401 — Authorization Required (требуется авторизация);
  - 403 — Forbidden (запрещено);
  - 404 — Not Found (не найдено);
  - 500 — Internal Server Error (внутренняя ошибка сервера).

  Список всех кодов состояния Apache приведен по адресу http://www.w3.org/Protocols/rfc2616/rfc2616-sec10.html.

  По умолчанию `ErrorDocument` не конфигурируется. Чтобы избежать некрасивых сообщений об ошибках, нужно заранее приложить усилия и создать страницу для ошибок, указав в `ErrorDocument`, где она находится.
- **Alias** — обеспечивает ассоциирование веб-каталога с любым каталогом с жесткого диска (также вне `DocumentRoot`). Например, с помощью `Alias /mytool/usr/`

local/mytool мы указываем, что при доступе к http://meinserver.de/mytool будут считываться файлы из каталога /urs/local/mytool.

Как правило, для каждого каталога alias из группы <Directory> нужно настраивать права доступа (подробнее об этом — в следующем подразделе). Для Alias существует вариант ScriptAlias, служащий для определения каталогов с помощью CGI-сценариев.

Конфигурация каталогов

Дополнительно к вышеописанным параметрам, которые действительны для всего виртуального хоста, можно настраивать свойства для отдельных каталогов вашего хоста в одной или нескольких группах <Directory "/*каталог*/">. В следующем списке перечислены важнейшие используемые при этом ключевые слова.

- **DirectoryIndex** — указывает, какой файл Apache должен отсылать, если адрес заканчивается на / и, таким образом, относится ко всему каталогу (по умолчанию это index.html). При этом можно указать и несколько файлов. В таком случае, Apache обрабатывает все введенные данные по порядку до тех пор, пока не будет найдено первое соответствие (например, DirectoryIndex index.php index.html).
- **Options** — позволяет указывать различные параметры, действующие для данного каталога. К таким параметрам относятся:
  - ExecCGI — позволяет выполнять сценарии CGI;
  - FollowSymLinks — обеспечивает переходы по символьным ссылкам;
  - Includes — добавляет включаемые файлы (модуль mod_include);
  - Indexes — показывает список файлов, если отсутствует index.html;
  - MultiViews — автоматический выбор языка (модуль mod_negotiation).

  По умолчанию в Apache действует настройка All. При этом активны все параметры за исключением MultiViews. Конфигурация Ubuntu предусматривает большее количество ограничений: для всей файловой системы действует Options FollowSymLinks, для каталога /var/www — Options Indexes FollowSymLinks MultiViews.

  Чтобы деактивировать отдельные параметры, по умолчанию заданные для вышестоящего каталога, перед ними нужно поставить знак «минус». Перед параметрами можно ставить и знак «плюс», но он не оказывает никакого действия: параметр с тем же успехом активизируется и без знака «плюс».

  Из соображений безопасности при применении параметров необходимо следовать правилу «чем меньше, тем лучше». Например, параметр Indexes позволяет любопытному пользователю узнать имена всех файлов, находящихся в каталоге — если вы забудете создать index.html. Разумеется, это может представлять угрозу. MultiView может потребоваться только на многоязычных сайтах, где автоматически выбирается язык. Если на вашем сайте такая возможность не предоставляется, от этого параметра также можно отказаться.
- **AllowOverride** — указывает, какие настройки, специфичные для конкретного каталога, могут быть изменены в файле .htaccess. На выбор предлагается:

- AuthConfig — настройка метода аутентификации;
- FileInfo — настройка типов файлов и документов;
- Indexes — модифицирование индекса каталогов;
- Limit — изменение прав доступа (Allow, Deny, Order);
- Options — изменение параметров каталога.

По умолчанию в Apache активны все возможности, то есть может быть изменен любой параметр. А в Ubuntu по умолчанию задается None (см. /etc/apache2/sitesavailable/default).

Защита каталогов

Права доступа к каталогам. Основного вопроса — управления правами доступа к каталогам — мы пока не затрагивали. В группе <Directory> можно с помощью Order, Allow и Deny указывать, при каких условиях Apache может читать и передавать файлы из конкретного каталога.

Правила доступ действуют и для всех подкаталогов, если в другой группе <Directory> явно не определены другие правила. В этом отношении правила доступа, заданные для каталога /, имеют особое значение. В данном случае речь идет о стандартных правилах доступа, действующих во всей файловой системе!

- **OrderAllow, Deny** — означает, что сначала интерпретируются все правила Allow, а потом — все правила Deny. Если к попытке доступа к сайту не применяются какие-либо правила, доступ блокируется.
- **Order Deny, Allow** — работает в обратном порядке по сравнению с предыдущим. Однако обратите внимание: если в таком случае к файлу не применимо ни одно из правил, доступ разрешается! Такой механизм действует в Apache по умолчанию.
- **Allowfrom** — указывает, с каких хост-имен, либо IP-адресов, разрешен доступ. Например: Allow from 213.214.215.216 izvestniysait.ru. Адресные пространства IP-адресов можно указывать в форме 213.214, или 213.214.0.0/255.255.0.0, или 213.214.0.0/16 (для 213.214.*.*). При задании хост-имен sait.ru приравнивается к www.sait.ru, sub.sait.ru и т. д. Правило Allow from all разрешает любой доступ.
- **Denyfrom** — работает прямо противоположным параметру Allowfrom образом и блокирует доступ к указанным хостам или, соответственно, адресам.

Еще одно замечание относительно настроек в /etc/apache2/sites-available/default: для корневого каталога / отсутствуют команды Order, Allow и Deny. В таком случае автоматически действует Order Deny,Allow, и при отсутствии каких-либо других команд, блокируется весь доступ ко всем каталогам! Исключениями являются лишь /var/www, /usr/lib/cgi-bin и /usr/share/doc, а также их подкаталоги. Если вы размещаете веб-файлы в других каталогах, то не забудьте разрешить к ним доступ.

Блокирование интернет-доступа. Если вы настраиваете Apache для обеспечения внутрикорпоративной коммуникации, то можете ограничить веб-доступ локальным каталогом. Если в локальной сети используется адресное пространство 192.168.1.* и локальный домен .sol, то верная конфигурация для /var/www выглядит так:

```
<Directory /var/www/>
  Options       Indexes FollowSymLinks MultiViews
  AllowOverride None
  Order         Deny,Allow
  Deny          from all
  Allow         from 192.168.1
  Allow         from localhost
  Allow         from .sol
</Directory>
```

Альтернативный метод заключается в том, чтобы задать с помощью `Listen` IP-адрес локального сетевого интерфейса (в Debian/Ubuntu это делается в файле `/etc/apache2/ports.conf`). В таком случае предполагается, что этот IP-адрес статический. Допустим, сервер имеет два сетевых интерфейса: один для связи с Интернетом, а другой — для связи с локальной сетью с IP-адресом 192.168.1.17. Таким образом, при указании `Listen 192.168.1.17` возникает ситуация, в которой Apache реагирует лишь на запросы, поступающие из локальной сети. В любом случае `Listen` действует для всей конфигурации Apache, а не только для отдельных каталогов или виртуальных хостов.

Третий вариант связан с применением брандмауэра. Брандмауэр должен препятствовать приему таких пакетов, которые приходят извне (из Интернета) и направлены на порты 80 и 443 (HTTPS). В принципе применение брандмауэра — хорошая идея, так как он функционирует совершенно независимо от Apache.

Защита веб-каталогов паролем. С точки зрения безопасности, использование Apache только в пределах локальной сети — случай идеальный, но такая ситуация имеет другой серьезный недостаток. Так мы исключаем из сети любых сотрудников, которые работают вне офиса. Альтернатива заключается в том, чтобы открывать доступ после прохождения аутентификации (то есть после указания имени пользователя и пароля). Для этого в Apache применяется очень простой метод, о котором мы и поговорим в данном пункте. Он также позволяет закрывать от нежелательных посещений конкретные каталоги. Например, на многих сайтах существует административный раздел, который предназначен только для людей, имеющих право изменять сайт и занимающихся его поддержкой.

Первый компонент при защите паролем — это файл пароля. Из соображений безопасности этот файл обязательно должен располагаться вне веб-каталога, чтобы к нему было невозможно попасть в результате запроса по веб-адресу (`http://servername/directory/passwordfile`). В следующем примере предполагается, что файл с паролем сохранен в каталоге `/var/www-private`. Создавая новый каталог, обращайте внимание на то, что Apache с аккаунтом `www-data` имеет право его читать.

Чтобы создать новый файл с паролем, воспользуйтесь командой `htpasswd` с параметром `-c` (create — создать). Разумеется, пароль шифруется.

```
root# cd /var/www-private
root# htpasswd -c passwords.pwd username
New password: ********
Re-type new password: ********
Adding password for user username
```

Следующие пары логин/пароль добавляются командой htpasswd без параметра -c:

```
root# cd /var/www-private
root# htpasswd passwords.pwd name2
New password: ********
Re-type new password: ********
Adding password for user username
```

Теперь мы имеем два варианта, позволяющих сконфигурировать Apache так, чтобы файл с паролем действительно учитывался системой. При первом варианте вы выполняете конфигурацию хоста непосредственно в конфигурационном файле Apache /etc/apache2/sites-available/default (для стандартного сайта сервера) или в -/name (для виртуального хоста). При применении второго варианта конфигурация производится в файле .htaccess, который располагается внутри веб-каталога.

Чтобы Apache учитывал файл пароля, нужно добавить различные параметры аутентификации в группу <Directory>. Если для каталога, который требуется защитить, пока отсутствует собственная группа <Directory>, создайте новую группу. При этом автоматически вступают в силу все параметры вышестоящего каталога. Таким образом, требуется самостоятельно добавить лишь параметры аутентификации. Образец показан в следующих строках:

```
# в /etc/apache2/sites-available/default (Ubuntu)
...
# Каталог, защищенный паролем
<Directory "/var/www/admin/">
   AuthType        Basic
   AuthUserFile    /var/www-private/passwords.pwd
   AuthName        admin"
   Require         valid-user
</Directory>
```

Кратко рассмотрим ключевые слова.

- **AuthType** — указывает тип аутентификации. Apache поддерживает два типа: Basic и Digest (второй вариант надежнее, но несовместим со старыми браузерами). Мы будем обсуждать тип Basic.
- **AuthUserFile** — задает местоположение файла с паролем.
- **AuthName** — указывает область (пределы), в которой может быть осуществлен доступ. Смысл в том, что вам не приходится каждый раз выполнять вход в систему, если вы обращаетесь к различным каталогам, которые защищены одним и тем же файлом пароля. Как только вы вошли в систему с определенным именем AuthName, этот вход является действительным и для всех остальных каталогов с таким именем AuthName.
- **Require valid-user** — означает, что для входа в систему подходит любая верная комбинация логина и пароля. В качестве альтернативы можно также указать, что в систему имеют право входить только определенные пользователи:

  ```
  Require user name1 name2
  ```

Описанный выше подход возможен лишь при условии, что у вас есть доступ к конфигурационным файлам Apache, то есть в случае, когда вы сами являетесь

веб-администратором. Если вы им не являетесь, то равноценную защиту можно обеспечить с помощью файла `.htaccess`, находящегося в защищаемом каталоге. В этом файле должны располагаться те же команды, что и в группе `<Directory>`, то есть `AuthType`, `AuthUserFile`, `AuthName` и `Require`.

ПРИМЕЧАНИЕ

Файлы .htaccess учитываются лишь в случае, если внутри каталога разрешено изменять информацию об аутентификации. (Вышестоящая) группа <Directory> должна содержать AllowOverride, AuthConfig или AllowOverride All.

23.3. Виртуальные хосты

Создание собственного сервера для каждого веб-сайта (хоста) — это излишняя трата ресурсов, особенно с учетом работоспособности современных компьютеров. Apache позволяет с помощью так называемых виртуальных хостов параллельно настраивать практически любое количество сайтов. Пока общее количество обращений не превысит возможностей компьютера, пользователи не заметят, что все эти сайты в действительности работают на единственном компьютере.

С технической точки зрения, Apache может тремя способами определять, на какой виртуальный хост будет направляться веб-запрос. В качестве исходной информации в любом случае используется HTTP-заголовок, передаваемый из браузера на сервер.

- **Виртуальные хосты, базирующиеся на именах.** По хост-имени, содержащемся в HTTP-заголовке, Apache определяет нужный веб-сайт. Этот вариант проще всего реализовать, и он наиболее распространен.
- **Виртуальные хосты, базирующиеся на IP-адресах.** По IP-адресу, содержащемуся в заголовке, Apache определяет нужный сайт. Но этот метод сопряжен с очень серьезным недостатком: каждый виртуальный хост требует собственного IP-адреса, а это, как правило, ресурс дефицитный. И все же есть причины, по которым стоит создавать хосты на базе IP: так, например, индивидуальный IP-адрес обязательно требуется, если данные сайта должны передаваться в зашифрованном виде (HTTPS). Если вы хотите реализовать на вашем сайте функции регулирования ширины полосы (либо если это необходимо), то вам скорее всего также потребуется собственный IP-адрес.
- **Виртуальные хосты, базирующиеся на портах.** Apache определяет нужный сайт по номеру порта. Этот вариант на практике применяется редко, так как в таком случае номер порта должен указываться в качестве части веб-адреса. Данный способ записи выглядит довольно запутанным и больше всего подходит для технически продвинутой целевой аудитории (например, для целей администрирования).

В этом разделе я снова буду исходить из стандартной конфигурации Debian и Ubuntu. В данных системах принято применять для каждого виртуального хоста собственный конфигурационный файл.

Если вы настраиваете веб-сервер в Fedora или RHEL, то, разумеется, при работе применяются те же ключевые слова Apache, которые были описаны выше. Так

или иначе, все настройки выполняются в центральном конфигурационном файле /etc/httpd/conf/httpd.conf. В конце этого файла находятся некоторые указания по определению виртуальных хостов.

Создание виртуальных хостов

В стандартной конфигурации Ubuntu даже стандартный веб-сайт (каталог /var/www) уже определен в /etc/apache2/sites-available/default как виртуальный хост. В конфигурационном файле /etc/ apache2/ports.conf, кроме того, содержится запись NameVirtualHost *:80. Это ключевое слово необходимо для того, чтобы Apache поддерживал хосты, основанные на именах. Символ * означает, что Apache интерпретирует в контексте виртуальных хостов все входящие запросы. Если на вашем сервере настроено несколько IP-адресов, но для хостов, основанных на именах, выделен лишь один такой адрес, то этот адрес нужно явно указать в NameVirtualHost.

Чтобы определить новый виртуальный хост, создайте в Debian и Ubuntu новый файл в каталоге /etc/apache2/sites-available/default. В трех следующих листингах приводится по одному примеру хостов, основанных, соответственно, на имени, IP-адресе и порте.

Виртуальные хосты, основанные на именах

При работе с виртуальными хостами, которые основаны на именах, адрес, указываемый в <VirtualHost>, должен соответствовать NameVirtualHost. ServerName задает имя хоста. Это имя должно содержаться в информации заголовка веб-запроса, чтобы Apache на него отреагировал! Вы также можете указать и другие имена с помощью ServerAlias. Например, при настройке рекомендуется дополнять ServerName www.mojserver.ru записью ServerAlias mojserver.ru, чтобы виртуальный хост мог использоваться как с указанием букв www, так и без них.

```
# /etc/apache2/sites-available/primer-named-host (Debian/Ubuntu)
<VirtualHost *:80>
 DocumentRoot   /var/directory1/
 ServerName     www.firma-1.ru
 ServerAlias firma-1.ru
 ...
</VirtualHost>
```

Виртуальные хосты, базирующиеся на IP-адресах и портах

При работе с IP-адресами, базирующимися на IP-адресах и портах, IP-адрес должен соответствовать IP-адресу сервера:

```
# /etc/apache2/sites-available/primer-IP-host  (Debian/Ubuntu)
<VirtualHost 213.214.215.216:80>
 DocumentRoot   /var/directory2/
 ServerName     www.firma-2.com
 ...
</VirtualHost>

# /etc/apache2/sites-available/primer-port-host    (Debian/Ubuntu)
<VirtualHost 213.214.215.216:12001>
```

```
 DocumentRoot   /var/directory3/
 ServerName     www.admin-firma3.ru
 ...
</VirtualHost>
```

Все остальные настройки в группе <VirtualHost> осуществляются точно так же, как и на стандартном сайте.

ПРИМЕЧАНИЕ

Информация об адресах и портах, указанная в <VirtualHost>, никак не влияет на то, какие адреса и порты отслеживает Apache. При стандартной конфигурации Ubuntu Apache обрабатывает запросы с любых адресов, поступающие на порты 80 и 443 (HTTPS). Соответствующая настройка осуществляется в /etc/apache2/ports.conf. Если Apache должен отслеживать и другие порты, то нужно необходимым образом дополнить файл ports.conf. Более подробная информация о конфигурации Apache для работы с виртуальными хостами находится по адресу http://httpd.apache.org/docs/2.2/ru/vhosts/.

Активизация виртуальных хостов

Чтобы активизировать (а позже — деактивировать) виртуальный хост, необходимо выполнить в Debian/Ubuntu, соответственно, команду a2ensite *имя* или a2dissite *имя*. В результате Apache заново загрузит конфигурационные файлы:

```
root# a2ensite beispiel-named-host    (Debian/Ubuntu)
root# service apache2 reload
```

Теоретически a2dissite позволяет деактивировать и стандартный сайт сервера. Но этого делать не следует, поскольку в файле /etc/apache2/sites-available/default находятся различные стандартные настройки для работы Apache!

Как только виртуальные хосты будут настроены, стандартный сайт, определенный в sites-available/default, будет отображаться лишь в случае, когда полученный запрос не будет относиться ни к одному из виртуальных хостов.

В Fedora и RHEL команды a2ensite/a2dissite не требуются, так как вся информация о виртуальных хостах находится в центральном конфигурационном файле httpd.conf. Для активизации сделанных там изменений применяется следующая команда:

```
root# service https reload  (RHEL, Fedora)
```

Пример

В этом подразделе описано, как создать на сервере Debian/Ubuntu новый виртуальный хост firma-123.ru — вместе с новым логином firma123, так, чтобы ваш друг, клиент и т. д. мог самостоятельно администрировать виртуальный хост. При этом я исхожу из того, что веб-файлы и файлы логов виртуального хоста будут находиться в домашнем каталоге нового пользователя firma123. С тем же успехом для этой цели можно создать новый каталог /var/www-firma123 и дать пользователю право записи информации в него.

Первый шаг заключается в том, чтобы создать новую учетную запись (аккаунт), присвоить ей пароль и создать необходимые каталоги. В следующих командах, разумеется, необходимо заменить firma123 на требуемое имя пользователя!

```
root# adduser firma123
root# passwd firma123
Enter new UNIX password: ********
Retype new UNIX password: ********
passwd: password updated successfully
root# mkdir                    ~firma123/www
root# chown firma123:firma123  ~firma123/www
root# mkdir           ~firma123/www-log
root# chown root:root ~firma123/www-log
root# chmod go-w      ~firma123/www-log
```

На втором этапе вы создаете новый файл в каталоге `sites-available`, который строится примерно так, как показано в следующем примере. Опять же `firma123` нужно заменить настоящим именем пользователя, а `firma-123.ru` — настоящим хост-именем. С помощью `AllowOverride AuthConfig File` вы предоставляете вашему клиенту относительно широкие возможности, позволяющие настраивать конфигурацию сайта в `.htaccess`. Если вас не устраивает такой вариант, то нужно обговорить различные детали конфигурации и жестко их задать.

```
# /etc/apache2/sites-available/firma-123.ru
<VirtualHost * >
  DocumentRoot     /home/firma123/www/
  ServerName       firma-123.ru
  ServerAlias      www.firma-123.ru
  ErrorLog         /home/firma123/www-log/error.log
  CustomLog        /home/firma123/www-log/access.log combined
  ServerAdmin      webmaster@firma-123.ru
  ErrorDocument    404 /not-found.html

  <Directory "/home/firma123/www/" >
    AllowOverride AuthConfig File
  </Directory>
</VirtualHost>
```

Для активизации сайта выполните следующие команды:

```
root# a2ensite firma-123.ru
root# service apache2 reload
```

Теперь клиент должен соответствующим образом настроить DNS-конфигурацию своего домена. Присваиваемый адрес должен соответствовать адресу, получаемому с вашего сервера. Если это так, то ваш веб-сервер будет реагировать на все запросы, поступающие по адресу www.firma-123.ru.

Права доступа. Как уже упоминалось выше, веб-функции Apache работают не с правами администратора, а с аккаунтом с ограниченными правами (`www-data` в Debian/Ubuntu, `apache` в Fedora/RHEL, `wwwrun` в SUSE). Это делается из соображений безопасности. Поэтому права доступа ко всем веб-файлам должны быть настроены так, чтобы Apache мог читать эти файлы.

Если Apache также должен изменять отдельные файлы (например, с помощью PHP-сценария), файлы и каталоги причисляются к группе `www-data/apache/wwwrun` и члены этой группы получают права записи (`chmod g+w`).

Журналирование (логирование). В предыдущем примере все сообщения об ошибках и о доступе сохраняются в собственных файлах виртуального хоста. Такой метод облегчает интерпретацию файлов логов. В любом случае количество общедоступных указателей файлов для Apache (как и для любого другого процесса Linux) ограничено. Если вы создаете очень много виртуальных хостов, то все обращения к ним нужно протоколировать в центральном файле и разбивать этот файл на более мелкие (соответствующие хостам) с помощью специальной программы. Подробнее об этом процессе рассказано на сайте http://httpd.apache.org/docs/2.2/vhosts/fd-limits.html.

Тест. Для работы с виртуальными хостами необходимо, чтобы DNS-конфигурация была верной! Чтобы протестировать сайт firma-123.ru, описанный в предыдущем примере, DNS-запись домена firma-123.ru должна указывать на IP-адрес вашего веб-сервера. Изменения в запись DNS может вносить только обладатель домена. В большинстве систем для обслуживания доменов для этого предоставляются специальные инструменты. Обратите внимание на то, что изменения DNS вступают в действие не сразу. Синхронизация многочисленных, распределенных по всему миру серверов имен может занимать несколько часов (хотя иногда это происходит и быстрее).

При реорганизации и переезде сервера часто требуется сначала спокойно протестировать новый сервер, а только потом действительно внести изменения в DNS. Простейший способ заключается в том, чтобы сначала сконфигурировать новый хост только на базе порта, а не на базе имени. Для этого удалите записи `ServerName` и `ServerAlias` и укажите в файле `<VirtualHost>` вместо звездочек IP-адрес сервера и номер порта, а также свободный номер порта, например, вот так:

```
<VirtualHost 213.214.215.216:12001 >
```

По умолчанию Apache обрабатывает лишь такие запросы, которые направляются к портам 80 и 443 (для HTTPS). Чтобы Apache учитывал и применяемый здесь порт 12001, в файле `/etc/apache2/ports.conf` нужно вставить еще одну строку — `Listen 12001`. Кроме того, требуется команда `service apache2 reload`, после которой Apache начнет работать с измененной конфигурацией. Теперь вы можете протестировать новый веб-сайт в браузере. При этом IP-адрес сервера задается вместе с номером порта 12001, например, вот так: http://213.214.215.216:12001.

23.4. Зашифрованные соединения

При обычном обмене данными между веб-сервером и браузером используется протокол HTTP. Он прост, передает все данные в незашифрованном виде (то есть текст остается открытым), поэтому совершенно не подходит для передачи такой информации, как, например, номера кредитных карт или другие личные данные. Для подобных целей применяется протокол HTTPS.

HTTPS объединяет «протокол передачи гипертекста» (HTTP) с уровнем защищенных сокетов (SSL). Таким образом, HTTP обогащается функциями шифрования. В этом разделе рассказано, как сконфигурировать Apache для работы с HTTPS-соединениями.

Сертификаты

Прежде чем переходить непосредственно к конфигурации, нужно создать серверный сертификат. И здесь потребуется небольшая вводная часть.

Основы. Шифрование данных осуществляется на основе ассиметричного шифрования (asymmetric encryption). Суть этого метода заключается в том, что существует пара ключей, причем один ключ в этой паре является общедоступным (открытым), а другой — закрытым (тайным). Открытый ключ пригоден только для шифрования данных. Чтобы их *рас*шифровать, нужен закрытый ключ. Детально описывать здесь этот метод я не буду — он уже многократно рассматривался и объяснялся, в частности, в «Википедии».

При установке соединения между клиентом (веб-браузером) и сервером (Apache) сначала (на основе случайного числа, предоставляемого со стороны клиента, и открытого ключа — с сервера) создается общий ключ. Этот процесс называется рукопожатием, или квитированием. Такой *ключ соединения* (session key) потом применяется при шифровании всей дальнейшей коммуникации в рамках этого соединения.

Но этот способ обеспечения безопасности, который, насколько известно на данный момент, практически полностью защищен от перехвата, является всего лишь *одним из* аспектов повышения безопасности. Не менее важный второй аспект заключается в том, что пользователь должен быть уверен, что связывается именно с тем адресатом, к которому обращался. Какая польза от шифрования банковской информации, если перехватить ее не удалось, но денежный перевод все равно поступил прямиком на счет мошенника?

Поэтому в серверном сертификате содержится не только открытый ключ сервера, но и информация о веб-сайте, а также своеобразная подпись сертифицирующей организации. Назначение этой подписи — проверять личность соискателя сертификата по копии паспорта, патенту и т. д. К сожалению, процесс такой проверки делает серверные сертификаты относительно дорогими.

Авторитетность конкретного сертификата зависит от авторитетности аутентифицирующего органа. Распространенные браузеры, например Internet Explorer или Firefox, принимают только такие сертификаты, которые были выданы авторитетными аутентифицирующими организациями (такими как Verisign или Thawte). При получении других сертификатов система выдает пространные предупреждения. Но, проявив некоторое упорство, можно заставить браузер принимать и сравнительно ненадежные сертификаты, но сделать на этой ниве цветущий онлайновый бизнес невозможно. Иными словами: для серьезных деловых сайтов авторитетный сертификат просто незаменим.

Создание самостоятельно подписанного сертификата. После того как я всеми силами попытался заверить вас в необходимости получения «правильного» сертификата, я расскажу, как вы сами можете создать сертификат. Для этого есть достаточно веские основания. Я уверен, что для первых экспериментов вполне достаточно самостоятельно созданного сертификата. Между прочим, на создание собственного сертификата достаточно всего нескольких минут, а вот на получение авторизованного сертификата у меня уходило иногда по нескольку дней, а то и недель. Чтобы потратить этот вынужденный период ожидания с пользой, стоит познакомиться с важнейшими «подводными камнями». Если в принципе все будет

работать, то вам не составит никакого труда заменить ваш самодельный сертификат настоящим.

Первым делом установите пакет `openssl`. В нем содержится одноименная команда, которая, в частности, позволяет создавать ключи и сертификаты. Соответственно:

```
root# apt-get install openssl (и) yum install openssl
```

Следующая команда создает файл, в котором находится закрытый и открытый ключи. Чтобы закрытый ключ не мог быть считан в виде незашифрованного текста, сам файл ключей также шифруется. Для этого вы должны указать специальную «фразу-пароль», то есть как можно более длинную, состоящую из нескольких слов последовательность символов. (Далее я буду пользоваться более устоявшимся термином «пароль», подразумевая, что речь идет о пароле из нескольких слов.)

```
root# openssl genrsa -des3 -out server.key 1024
...
Enter pass phrase for server.key: ******************
Verifying – Enter pass phrase for server.key: ******************
```

В дальнейшем всякий раз, когда вы пожелаете воспользоваться закрытым ключом, указывайте пароль. Поскольку закрытый ключ нужен и для Apache, такой пароль вводится также при каждом запуске Apache. При работе с сервером подобный вариант неприемлем. Поэтому следующая команда снимает шифрование. Теперь в `server.csr` и открытый, и закрытый ключи записаны открытым текстом. Обратите внимание: читать этот файл может только администратор (root). Если этот файл попадет в чужие руки, сертификат сервера моментально потеряет всякую ценность и его потребуется отозвать!

```
root# openssl rsa -in server.key -out server.pem
Enter pass phrase for server.key: ******************
writing RSA key
root# chmod 400 server.pem
```

Чтобы завершить создание сертификата, нужно проделать еще кое-какую работу. Нужно указать, в какой стране и по какому адресу вы живете, как вас зовут и т. д. Здесь очень важно правильно ответить на вопрос об общем имени (Common Name). В данном случае имеется в виду не имя, а точное название вашего сайта — такое, которое будет использоваться при создании зашифрованных соединений. Некоторые сайты применяют для зашифрованных соединений собственный поддомен (например, banking.ing-diba.de[1]), другие — нет (www.amazon.de). Как обычно, сертификат действителен только при правильном способе записи. Например, вы не можете использовать сертификат www.firma-abc.de для работы с фирмой firma-abc.de (и наоборот!).

В сертификате обычно оказывается и открытый ключ, извлекаемый командой `openssl` из `server.key`.

```
root# openssl req -new -key server.key -out server.csr
Enter pass phrase for server.key: ******************
You are about to be asked to enter information that will be
incorporated into your certificate request.
```

[1] Страница онлайн-банкинга немецкого банка. — *Примеч. ред.*

```
What you are about to enter is what is called a Distinguished
Name or a DN. There are quite a few fields but you can leave
some blank. For some fields there will be a default value,
If you enter '.', the field will be left blank.
-----
Country Name (2 letter code) [AU]:              DE
State or Province Name (full name) [...]:       .
Locality Name (eg, city) []:                    Berlin
Organization Name (eg, company) [Sample Ltd]:   Max Muster
Organizational Unit Name (eg, section) []:      .
Common Name (eg, YOUR name) []:                 www.firma-abc.de
Email Address []:                               webmaster@firma-abc.de

Please enter the following 'extra' attributes
to be sent with your certificate request
A challenge password []:                        .
An optional company name []:                    .
```

С помощью следующей команды вы сами подписываете ваш сертификат. В случае с настоящим сертификатом этот процесс (после проверки предъявленных вами документов) происходит в аутентифицирующей организации. Для подписывания сертификата при этом используется ключ данной организации.

По умолчанию готовый сертификат остается действительным в течение 30 дней. Параметр `-days 1900` удлиняет срок действия сертификата примерно до пяти лет.

```
root# openssl x509 -req -days 1900 -in server.csr -signkey server.key \
                   -out server.crt
Signature ok
subject=/C=DE/L=Berlin/O=Max Muster/CN=www.firma-abc.de/
   emailAddress=webmaster@firma-abc.de
Getting Private key
Enter pass phrase for server.key: ******************
```

С физической точки зрения, созданные ключи и сертификаты — это сравнительно небольшие текстовые файлы. Чтобы отобразить открытым текстом данные, содержащиеся в сертификате, пользуйтесь командой `openssl x509 -text`. Следующий вывод сокращен ради экономии места:

```
root# ls -l
... 696 ... server.csr      (Сертификат без подписи)
... 963 ... server.key      (Зашифрованный закрытый ключ)
... 887 ... server.pem      (Незашифрованный закрытый ключ)
... 936 ... server.crt      (Сертификат с подписью)
root# cat server.crt
-----BEGIN CERTIFICATE-----
MIICWTCCAcICCQCL6ExhrQiELDANBgkqhkiG9w0BAQUFADBxMQswCQYDVQQGEwJB ...
-----END CERTIFICATE-----
root# openssl x509 -text -in server.crt
Certificate:
   Data:
      Version: 1 (0x0)
      Serial Number:
```

```
        8b:e8:4c:61:ad:08:84:2c
    Signature Algorithm: sha1WithRSAEncryption
    Issuer: C=DE, L=Berlin, O=Max Muster, CN=www.firma-abc.de/
          emailAddress=webmaster@firma-abc.de
    Validity
        Not Before: Oct 28 14:03:18 2011 GMT
        Not After : Jan 10 14:03:18 2017 GMT
    Subject: C=DE, L=Berlin, O=Max Muster, CN=www.firma-abc.de/
           emailAddress=webmaster@firma-abc.de
    Subject Public Key Info:
        Public Key Algorithm: rsaEncryption
        RSA Public Key: (1024 bit)
            Modulus (1024 bit):
                00:ed:1f:08:cb:f4:4d:ef:a6:f6:0a:be:b3:c2:92: ...
            Exponent: 65537 (0x10001)
Signature Algorithm: sha1WithRSAEncryption
    05:46:af:5c:12:84:28:59:e4:8f:db:2d:2d:0f:4a:3c:0e:84: ...
```

Бесплатные сертификаты. Отдельные компании, например https://www.startssl.com/ или cheapssls.com, предлагают бесплатные или очень дешевые сертификаты. В таком случае посетители вашего сайта могут быть уверены, что действительно связываются с http://vash-website.ru, а не с другим хостом. При работе с такими сертификатами происходит лишь валидация домена. Не проверяется, кто в действительности стоит за этим сайтом. Соответственно, любой злоумышленник может без труда приобрести такой сертификат. Еще один недостаток таких сертификатов — ограниченный срок действия (не более одного года в случае с StartSSL).

Несмотря на эти ограничения, сертификаты таких фирм, как StartSSL или CheapSSL, часто предпочитаются самостоятельно изготовленному сертификату. Большинство браузеров принимает такие сертификаты, не выдавая никаких тревожных предупреждений. Но степень надежности, отмечаемая в браузере для данного сайта, невелика (например, в списке адресов в данном случае не будет показана зеленая полоска с названием компании). Тем не менее, зашифрованный обмен данными при работе с такими сертификатами в большинстве браузеров осуществляется без проблем.

Поэтому бесплатные или дешевые доменные сертификаты представляют собой компромисс между самостоятельным изготовлением сертификатов и прохождением процедуры проверки подлинности. Но не думайте, что создать файлы сертификатов проще, чем с помощью команды `openssl`. Как раз сайт StartSSL особенно неудобен в использовании. Чтобы обеспечить надежность связи между StartSSL и браузером, на вашем компьютере устанавливается собственный ключ. Этот ключ является необходимым элементом для каждого последующего входа в систему. По возможности используйте Firefox. По моему опыту, именно в этом браузере с StartSSL возникает наименьшее количество проблем.

Конфигурация Apache для работы с HTTPS

Mod_ssl. Функции Apache, необходимые для работы с протоколом HTTPS, находятся в модуле `mod_ssl`. В Debian и Ubuntu этот модуль устанавливается по умолчанию, его нужно только активизировать.

```
root# a2enmod ssl
root# service apache2 restart
```

В Fedora и RHEL сначала нужно установить SSL-модуль:

```
root# yum install mod_ssl
root# service httpd restart
```

Виртуальный хост HTTPS. Apache должен считывать файл с ключом и файлом сертификата. Поэтому вполне логично скопировать оба этих файла в конфигурационный каталог Apache:

```
root# cp server.pem server.crt /etc/apache2
```

Далее нужно дополнить `httpd.conf` (Fedora, RHEL) записью `VirtualHost` либо вставить соответствующие строки в новый файл `/etc/apache2/sites-available`. В Debian и Ubuntu соответствующий файл шаблона сразу имеется в дистрибутиве (`default-ssl`). Можете использовать данный файл в качестве исходного материала для собственного файла. Если вы хотите, чтобы он заметнее отличался от других подобных файлов, поставьте перед ним `ssl` или `https` (например, `ssl.firma-abc.de`).

В приведенных ниже строках показана минимальная конфигурация, при которой параллельно создаются и стандартный сайт (HTTP), и сайт для работы с HTTPS. Для обоих сайтов применяется один и тот же IP-адрес. Разница заключается только в номере порта (443 для HTTPS, см. строку `VirtualHost`).

`SSLEngine on` активизирует функции шифрования. `SSLxxxFile` указывает, где находятся файлы с сертификатом и с закрытым ключом. `SSLProtocol` и `SSLCipherSuite` определяют, какая версия SSL-протокола будет применяться и, соответственно, какой механизм будет использоваться для создания общих ключей соединений. Как правило, Apache и браузер обмениваются информацией о том, какие протоколы они поддерживают, и задействуют наиболее безопасный метод, которым располагают. Только при наличии достаточно веских причин — например, если для вас неприемлемы определенные старые методы и протоколы — здесь нужно специально задать желаемые значения по умолчанию.

```
# Например, в /etc/httpd/conf/httpd.conf (Fedora/RHEL)
# или в /etc/apache2/sites-available/ssl.firma-abc.de
<VirtualHost _default_:443>
 ServerName           www.firma-abc.de
 DocumentRoot         /var/www/
 SSLEngine            on
 SSLCertificateFile   /etc/apache2/server.crt
 SSLCertificateKeyFile/etc/apache2/server.pem
 SSLProtocol          all -SSLv2
 SSLCipherSuite       HIGH:MEDIUM
 <Directory /var/www/>
  AllowOverride    None
  Order            allow,deny
  Allow            from all
 </Directory>
</VirtualHost>
```

Для активизации HTTPS-сайта нужно приказать Apache заново считать конфигурацию. Если конфигурация HTTPS-версии сайта в Debian/Ubuntu осуществлялась в собственном конфигурационном файле в `/etc/apache2/sites-available`, то этот файл нужно активизировать:

```
root# service httpd restart        (Fedora/RHEL)
root# a2ensite ssl.firma-abc.de    (Debian/Ubuntu)
root# service apache2 restart      (Debian/Ubuntu)
```

Если вы впервые переходите к работе с SSL, а раньше не пользовались этим механизмом, обычного повторного считывания конфигурационного файла недостаточно. Чтобы SSL-модуль загрузился, нужно указывать `restart`, а не `reload`!

Файлы SSLCertificateChainFile и SSLCACertificateFile. Ключевые слова `SSLCertificate`, применявшиеся в обоих приведенных выше вариантах конфигурации Apache, достаточны для работы как с самостоятельно созданными сертификатами, так и с сертификатами крупных признанных организаций (Thawte, VeriSign).

Если же вы используете сертификат организации, которая по умолчанию неизвестна браузеру, придется сообщить браузеру дополнительную информацию о том, как он может проверить ваши сертификаты. Это приходится делать в случаях работы со сравнительно небольшими сертифицирующими организациями, в частности — с StartSSL. Если быть точным, в данном случае речь идет об информации, какая общепризнанная сертифицирующая организация отвечает той сертифицирующей организации, с которой работаете вы. Браузер должен восстановить «цепь доверия» до сертифицирующей организации, которая ему известна. Если вы забудете эти дополнительные функции, то браузер будет «жаловаться» при посещении вашего сайта на «недоверенное соединение», поскольку браузеру не удалось восстановить цепочку до надежного сертифицирующего органа.

В таком случае вам помогут ключевые слова `SSLCertificateChainFile` и `SSLCACertificateFile`, с помощью которых вы передаете Apache сертификаты вашего сертифицирующего органа. Необходимые для сертификации файлы предоставляются для загрузки на сайте сертифицирующей организации. После того как вы настроите эти файлы на вашем компьютере так, что Apache сможет их читать, нужно изменить конфигурационный файл следующим образом и выполнить `apache2/httpd reload`.

```
...
SSLCertificateFile       /etc/apache2/server.crt
SSLCertificateKeyFile    /etc/apache2/server.pem
SSLCertificateChainFile  /etc/apache2/sub.class1.server.ca.pem
SSLCACertificateFile     /etc/apache2/ca.pem
...
```

Благодаря этой дополнительной информации, браузер сможет проверять корректность используемых вами сертификатов.

Несколько HTTPS-сайтов. Существенный недостаток HTTPS заключается в том, что при работе с несколькими HTTPS-сайтами для каждого из них нужен собственный IP-адрес. IP-адреса необходимо явно указать вместе с ключевым словом `VirtualHost`. Apache контролирует выполнение этого условия уже на этапе считывания конфигурационных файлов и предупреждает вас в случае возникновения конфликта адресов.

```
<VirtualHost 1.2.3.4:443>
 ServerName        www.noch-eine-firma.de
 DocumentRoot      /var/www-secure
 SSLEngine         on
 ...
</VirtualHost>
```

23.5. Awstats и Webalizer

Если у вас есть сайт, то вам, как правило, важно знать, сколько у вас посетителей. Две наиболее популярные программы для сбора статистических данных о доступе к сайту называются Awstats и Webalizer. Они интерпретируют файлы логов Apache. Результаты такой интерпретации потом можно просмотреть в браузере.

Сложно порекомендовать одну из этих программ. В целом, программа Awstats более современная. Она дает более подробные и зачастую более точные результаты. С другой стороны, ее достаточно сложно конфигурировать, а при стандартной настройке сохраняется довольно высокий риск с точки зрения безопасности.

По сравнению с Awstats, Webalizer можно назвать одним из первоэлементов всего Веба. Эта программа пока поддерживается, но уже много лет не развивается. Своей непреходящей популярностью Webalizer обязан, в первую очередь, простоте — как в конфигурации, так и в представлении результатов.

ПРИМЕЧАНИЕ

Если вы попробуете одновременно использовать обе программы и выбрать из них лучшую методом сравнения, то обнаружите, что они дают различные результаты. И та и другая программа с большим или меньшим успехом пытается отсеять нерелевантные результаты. Можно с упоением спорить о том, какой результат «правильнее». Но суть в том, что оба инструмента учитывают массу актов доступа, исходящих от поисковых роботов и автоматизированных сценариев загрузки.

Альтернатива. Вместо Awstat и Webalizer можно воспользоваться свободно распространяемой системой Piwik или Google Analytics (http://www.google.com/intl/de/analytics/). Для этого нужно внедрить в страницы вашего сайта код на JavaScript, переадресующий каждый акт доступа к странице на центральный сервер. Такой метод помогает лучше отличать реальных посетителей от поисковых роботов, и результат получается более точным.

Awstats

Основная идея Awstats заключается в том, что программа регулярно (например, раз в час) интерпретирует файлы логов Apache. Результаты записываются в текстовом формате в обычную базу данных `/var/lib/awstats`. Графическое и статистическое представление данных, как правило, осуществляется динамически, с помощью сценария CGI-Perl, который должен вызываться по адресу по следующему образцу:

- http://vash-website.ru/awstats/awstats.pl;
- http://vash-website.ru/awstats/awstats.pl?config=vtoraja-stranica.ru.

В принципе страницы результатов можно генерировать и ежедневно, с помощью Cron-сценария. Это более надежный вариант, однако он менее удобен для наблюдателя.

Awstats не начинает работать сразу после установки пакета `awstats`! Предварительно нужно создать по конфигурационному файлу Awstat для каждого виртуального хоста, включить в конфигурацию Apache статические страницы Awstat или CGI-сценарий Awstat. Кроме того, придется выполнить некоторую настройку, специфичную для конкретных дистрибутивов.

Конфигурация Awstats

Пока я работал над этим разделом, мне, к сожалению, показалось, что различий в конфигурации Awstats между Debian/Ubuntu и Fedora/RHEL больше, чем сходства. Но «общий знаменатель» присутствует: Awstats требует, чтобы для каждого виртуального хоста создавался собственный конфигурационный файл в каталоге `/etc/awstats`. Имя такого файла строится по образцу `awstats.`*`имя хоста`*`.conf`, например `awstats.firma123.ru.conf`. В качестве исходного пункта при создании новых конфигурационных файлов используйте файл-шаблон `awstats.conf` (Debian/Ubuntu) или `awstats.model.conf` (Fedora/RHEL):

```
root# cd /etc/awstats
root# cp awstats.conf awstats.firma123.ru.conf        (Debian/Ubuntu)
root# cp awstats.model.conf awstats.firma123.ru.conf  (Fedora/RHEL)
```

Каждый конфигурационный файл состоит примерно из 150 строк, но содержит почти исключительно одни комментарии. Как правило, приходится внести всего три изменения: один раз изменить место доступа к файлу логов и два раза — имя хоста:

```
# Изменения в /etc/awstats/awstats.<hostname>.conf
...
LogFile="/home/firma123/www-log/access.log"
SiteDomain="firma123.ru"
HostAliases="localhost 127.0.0.1 firma123.ru"
```

По умолчанию Awstats обрабатывает данные о доступе только за один месяц. Если вы хотите изучить данные за целый год, то потребуется внести еще одно изменение:

```
# Изменения в /etc/awstats/awstats.<hostname>.conf
...
AllowFullYearView=3
```

При этом, однако, учитывайте, что создание годовой статистики может происходить только динамически (статические результаты всегда сводятся лишь за месяц). Кроме того, сценарий Awstats при работе потребляет достаточно много памяти и ресурсов процессора.

В Debian и Ubuntu вам, наконец, придется переименовать файл-шаблон, чтобы этот файл не обрабатывался как активный конфигурационный файл:

```
root# mv /etc/awstats/awstats.conf /etc/awstats/awstats.conf.orig (Debian/
                                                                  Ubuntu)
```

Если вы хотите изменить некоторые общие настройки для *всех* конфигурационных файлов Awstats, пользуйтесь специально предусмотренным для этого файлом /etc/awstats/awstats.conf.local.

Awstats в Debian/Ubuntu

В Debian/Ubuntu файл /etc/cron.d/awstats отвечает за то, чтобы раз в десять минут выполнялся сценарий /usr/share/awstats/tools/update.sh. Этот сценарий, в свою очередь, вызывает применительно ко всем файлам /etc/awstats/awstats.*conf сценарий awstats.pl, написанный на Perl, и, таким образом, обновляет статистическую базу данных Awstats. Чтобы файл-шаблон awstats.conf не расценивался при этом как активный конфигурационный файл, его нужно переименовать (см. выше).

Сначала доступ к файлам логов Apache не удается выполнить из-за несовместимости прав доступа. Ведь файлы логов доступны для чтения только администратору (root) и членам группы adm. Сценарии Awstats, напротив, выполняются под аккаунтом www-data.

Если поискать в Интернете по запросу ubuntu|debian awstat permission denied (ubuntu|debian awstat доступ запрещен), то найдете множество предложений по решению этой проблемы, но многие из них не выдерживают критики с точки зрения безопасности. Наименьшее количество таких проблем возникает при изменении групповой принадлежности файлов логов Apache. С помощью двух следующих команд вы можете отнести каталог /var/log/apache2 и все содержащиеся в нем файлы к группе www-data и гарантировать, что новые создаваемые в этом каталоге файлы будут принадлежать к этой же группе.

```
root# chgrp -R www-data /var/log/apache2/
root# chmod 2755 /var/log/apache2/
```

Кроме того, необходимо убедиться, что с этого момента и logrotate будет использовать группу www-data. Для этого измените одну строку в конфигурации logrotate для Apache:

```
# в файле /etc/logrotate.d/apache2
...
create 640 root www-data
```

Наконец, необходимо позаботиться о том, чтобы Awstats интерпретировала файлы логов прежде, чем logrotate их переименует:

```
# в файле /etc/logrotate.d/apache2
...
prerotate
   if [ -x /usr/share/awstats/tools/update.sh ]; then
      su -l -c /usr/share/awstats/tools/update.sh www-data
   fi
endscript
```

Когда будет налажена автоматическая интерпретация файлов логов, вы, конечно, захотите просмотреть и результаты этой работы. По умолчанию Cron-сценарий /etc/cron.d/awstats раз в день создает статическую версию файлов с результатами. Они сохраняются в каталоге /var/cache/awstats/*имядомена*. Чтобы этот каталог мож-

но было просмотреть в браузере, добавьте следующие строки в файл виртуального хоста Apache:

```
# в /etc/apache2/hosts-available/firma123.ru
...
Alias      /awstatic        "/var/cache/awstats/kofler.info"
<Directory /var/cache/awstats/kofler.info>
   Options +Indexes
</Directory>
```

Для каждого года и каждого месяца Awstats создает собственные каталоги. Но центральный стартовый файл отсутствует. Чтобы просмотреть интерпретацию результатов за определенный месяц, в браузере нужно указать следующий адрес:

```
http://firma123.ru/awstatic/2012/09/awstats.firma123.ru.en.html
```

Параметр `+Indexes` обеспечивает удобную навигацию по каталогам с результатами и избавляет вас от утомительного ввода всего адреса целиком. Чтобы не случилось так, что каждый пользователь сайта firma123.ru, который угадает этот адрес, мог читать статистику, вы можете защитить данный каталог паролем.

За динамическое представление результатов Awstats отвечает CGI-сценарий `awstats.pl`. Чтобы Apache мог выполнять этот сценарий, в конфигурационный файл Apache нужно добавить следующие строки:

```
# в /etc/apache2/sites-available/default
...
Alias        /awstatsclasses "/usr/share/awstats/lib/"
Alias        /awstats-icon/  "/usr/share/awstats/icon/"
Alias        /awstatscss     "/usr/share/doc/awstats/examples/css"
ScriptAlias  /awstats/       "/usr/lib/cgi-bin/"
```

Результаты работы Awstats можно просмотреть на страницах:

- http://hostname/awstats/awstats.pl;
- http://hostname/awstats/awstats.pl?config=vtoraja-stranica.de.

Как и в случае со статической конфигурацией, динамическую конфигурацию также нужно защищать паролем. В данном случае защита распространяется на каталог `cgi-bin`. Просто добавьте соответствующие команды `AuthXxx` в блок `Directory` файла `/etc/apache2/sites-available/` для каталога `/usr/lib/cgi-bin`!

Более подробная информация о конфигурации Awstats для Debian и Ubuntu содержится в файле: /usr/share/doc/awstats/README.Debian.gz.

Характерные черты Fedora/RHEL

Хотя в Fedora предоставляется пакет для работы с Awstats, в RHEL такой пакет не дается. Соответственно, перед установкой приходится создать дополнительный источник пакетов. В таком качестве хорошо использовать EPEL (дополнительные пакеты для Enterprise Linux).

При работе с Awstats в Fedora/RHEL Cron-файл `/etc/cron.hourly/awstats` отвечает за то, чтобы один раз в час выполнялся сценарий `/usr/share/awstats/tools/awstats_updateall.pl`. В свою очередь этот сценарий обновляет базы данных Awstats

для всех конфигурационных файлов /etc/awstats/awstats.*conf, причем файл-шаблон awstats.model.conf не учитывается.

При установке Awstats сценарии из этого пакета пытаются автоматически настроить подходящие конфигурационные файлы для хоста. Но это удается лишь в случае, если вы не изменяли стандартное место сохранения файлов логов. Если в конфигурации Apache содержатся настройки, касающиеся других виртуальных хостов, вы должны будете самостоятельно настроить соответствующие необходимые конфигурационные файлы Awstats.

В отличие от ситуации с Debian/Ubuntu, в данном случае Awstats без каких-либо проблем получает доступ к файлам логов Apache. В Fedora и RHEL эти файлы, как правило, открыты для чтения каждому.

Чтобы программа logrotate не переименовывала файлы прежде, чем Awstats их интерпретирует, следует синхронизировать работу обеих служб. Для этого добавьте в файл /etc/logrotate.d/httpd следующие строки prerotate:

```
# в /etc/logrotate.d/httpd
...
   prerotate
      /usr/bin/awstats_updateall.pl now -configdir=/etc/awstats/ \
   --awstatsprog=/var/www/awstats/awstats.pl >/dev/null
endscript
```

Для доступа к создаваемым динамически интерпретациям Awstats предусмотрен конфигурационный файл /etc/httpd/conf.d/awstats.conf. При этом страницы с результатами находятся по следующим адресам:

- http://hostname/awstats/awstats.pl;
- http://hostname/awstats/awstats.pl?config=vtoraja-stranica.de.

Но из соображений безопасности сначала эти страницы доступны только для просмотра с localhost. Чтобы изменить эту ситуацию, замените Allow from 127.0.0.1 на Allow from all. Таким образом, вы, конечно, открываете каталог Awstats для просмотра кому угодно, а это, как правило, нежелательно. Поэтому в блоке Directory нужно указать для каталога /usr/share/awstats/wwwroot обычные записи AuthXxx, чтобы этот каталог можно было защитить паролем.

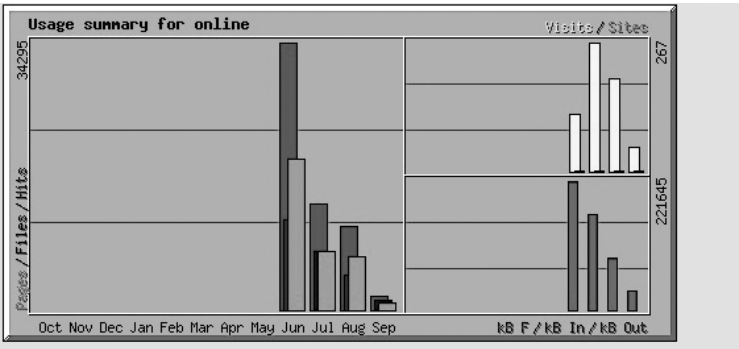
```
# в /etc/httpd/conf.d/awstats.conf
...
<Directory "/usr/share/awstats/wwwroot">
   Options None
   AllowOverride   None
   Order allow,deny
   Allow from all
   AuthType        Basic
   AuthUserFile    /etc/httpd/awstat-passwords.pwd
   AuthName        "awsuser1 awuser2 ..."
   Require         valid-user
</Directory>
```

Создание статических страниц с результатами Awstats в Fedora/RHEL не предусмотрено. Если из соображений безопасности вы хотите обойтись без динамиче-

ского просмотра статистики Awstats, нужно создать Cron-файл `/etc/cron.daily`, вызывающий сценарий `/usr/share/awstats/tools/awstats buildstaticpages.pl` на языке Perl. Параметры этого сценария документированы по следующей ссылке: http://awstats.sourceforge.net/docs/awstats_tools.html.

Webalizer

До сих пор наиболее популярной альтернативой Awstats остается программа Webalizer. По принципу работы она напоминает Awstats: как правило, Webalizer раз в день выполняется сценарием Cron, интерпретирует файлы логов Apache и после этого создает несколько статических страниц с результатами интерпретации (рис. 23.2). В отличие от случая с Awstats, динамическое представление результатов не предусмотрено. Таким образом, множество проблем, связанных с безопасностью, в данном случае просто исключено.



Summary by Month

Month	Daily Avg				Monthly Totals							
	Hits	Files	Pages	Visits	Sites	kB F	kB In	kB Out	Visits	Pages	Files	Hits
Sep 2008	347	241	186	10	1	32626	0	0	50	932	1208	1739
Aug 2008	346	146	219	6	1	89077	0	0	191	6795	4543	10741
Jul 2008	438	243	242	8	1	164478	0	0	267	7522	7552	13599
Jun 2008	1633	551	922	5	1	221645	0	0	118	19382	11572	34295
Totals						507825	0	0	626	34631	24875	60374

Рис. 23.2. Статистика обращений к сайту в программе Webalizer

Во всех распространенных дистрибутивах Webalizer предоставляется в готовом пакете:

```
root# apt-get install webalizer    (Debian/Ubuntu)
root# yum install webalizer        (Fedora/RHEL)
```

Конфигурация в Debian/Ubuntu

Конфигурация данной программы значительно отличается в зависимости от дистрибутива. В Debian и Ubuntu для управления Webalizer применяются файлы `*.conf`

в каталоге /etc/webalizer. В качестве примера можно использовать хорошо документированный файл-шаблон webalizer.conf.

Если на вашем сервере вы задействуете единственный сайт, а он применяет стандартные файлы логов, то никаких изменений вносить не требуется. В таком случае Webalizer интерпретирует файл /var/log/apache2/access.log.1 и сохраняет результаты в каталоге /var/www/webalizer.

Возможно, вас озадачит то, что Webalizer интерпретирует файл access.log.1, а не более новый файл access.log. Но преимущество данной ситуации заключается в том, что между Webalizer и logrotate не возникает конфликт. (Cron выполняет logrotate раньше, чем Webalizer). В любом случае logrotate переименовывает файлы логов Apache всего раз в неделю. Поэтому статистика Webalizer может запаздывать на неделю.

Простейшее решение такой проблемы заключается в следующем: можно сконфигурировать logrotate так, чтобы файлы логов переименовывались ежедневно (daily вместо weekly в /etc/logrotate.d/apache2).

Другое решение — все же обращаться в файле webalizer.conf к access.log. Но в данном случае необходимо гарантировать с помощью Incremental=yes, чтобы при каждом прогоне Webalizer запоминал, до какой степени он успел прочитать файл логов.

```
# в /etc/webalizer/webalizer.conf
...
LogFile=/var/log/apache2/access.log
Incremental=yes
```

Одновременно вам придется позаботиться и о том, чтобы logrotate выполнял Webalizer до того, как файл лога будет переименован:

```
# в /etc/logrotate.d/apache2
...
prerotate
   /etc/cron.daily/webalizer
endscript
```

Доступ к статистике Webalizer в браузере осуществляется по следующему адресу: http://hostname/webalizer/.

Чтобы любой желающий не мог прочитать статистическую информацию вашего веб-сервера, необходимо защитить паролем каталог /var/www/webalizer, содержащий статистическую информацию Webalizer.

Если вы настроили на вашем сервере несколько виртуальных хостов, то для каждого хоста, на котором есть собственный файл логов, нужно создать и собственный конфигурационный файл Webalizer. Следующий пример описывает ситуацию с сайтом firma-123.ru, файлы логов которого сохраняются в каталоге пользователя firma123.

```
# Файл /etc/webalizer/firma-123.ru.conf

# Основные настройки
LogFile      /home/firma123/www-log/access.log
OutputDir    /home/firma123/www/webalizer
```

```
# Маркировка
ReportTitle  Статистика доступа firma-123.ru
HostName     firma-123.ru

# Количество записей Top-n
TopURLs      50
TopKURLs     30
TopReferrers 50

# Не учитывать ссылку в списке Top-n
HideURL         *.gif
HideURL         *.jpg
HideURL         *.png

# Игнорировать обращения с localhost, а также ссылки с firma-123.ru
IgnoreSite      localhost
IgnoreReferrer  localhost
IgnoreReferrer  firma-123.ru
IgnoreReferrer  www.firma-123.ru
```

Еще несколько замечаний по данной конфигурации. Параметры `TopXxx` увеличивают количество записей в списках `Top n ... URLs`, `Top n ... URLs by kByte` и `Top n ... referrers`. Эти списки я нахожу очень интересными. Строки `HideURL` гарантируют, что в списках `Top n` не будут учитываться файлы изображений. Благодаря строкам `Ignore` обращения с локального компьютера, а также перекрестные ссылки с вашего собственного сайта учитываться также не будут.

Чтобы Webalizer мог сохранять свои результаты, необходимо, конечно же, создать каталог `/home/firma123/www/webalizer`:

```
root# mkdir /home/firma123/www/webalizer
root# chown firma123:firma123 /home/kofler/www/webalizer
```

Пользователь, отвечающий за работу сайта (в данном случае это `firma123`), должен дополнительно защитить доступ к сайту с помощью файла `.htaccess`.

СОВЕТ

Webalizer может сформировать список Top n ... referrers только при условии, что в файле логов Apache содержится соответствующая информация. Для этого в конфигурационном файле Apache /etc/apache2/ sites-available/* в строке CustomLog нужно задать форматирование combined (а не common)!

Конфигурация в Fedora/RHEL

В Fedora/RHEL Webalizer интерпретирует только конфигурационный файл `/etc/webalizer.conf`. Результаты работы сохраняются в каталоге `/var/www/usage` и, соответственно, их можно просмотреть по следующему адресу: http://hostname/usage.

Если вам требуется раздельная статистика по нескольким виртуальным хостам, то нужно не только самостоятельно создать соответствующие конфигурационные файлы Webalizer, но и сценарии Cron для вызова Webalizer.

Значительно проще, чем в Debian и Ubuntu, решается ситуация с устранением конфликтов между Webalizer и `logrotate`. Благодаря двум нулям, поставленным

перед webalizer в /etc/cron.daily/00webalizer, Webalizer всегда выполняется *раньше*, чем logrotate. (Cron всегда выполняет сценарии из cron.daily в алфавитном порядке.)

23.6. PHP

Динамические веб-страницы. Сам Apache может передавать только статические веб-страницы. Но на всех современных сайтах используются и динамические веб-страницы. Всякий раз при запросе такой страницы Apache запускает внешнюю программу, перерабатывает код страницы и отправляет в ответ страницу, которая отдельно настраивается для отображения в браузере (например, на странице может быть указано точное время или результат обработки запроса, посланного в базу данных, либо постоянно переключающиеся рекламные баннеры и т. д.).

PHP. Для программирования динамических веб-страниц применяется несколько языков, например Perl или Java. Но в мире Linux/UNIX наиболее популярен язык PHP — *препроцессор гипертекста*. Подробная информация о PHP приводится на сайте http://www.php.net/.

Основная идея создания сайтов на PHP заключается в том, что в файле с расширением *.php содержится код, написанный как на HTML, так и на PHP. Код PHP начинается тегом <?php и завершается тегом ?>. Если пользователь запрашивает в Интернете страницу, на которой есть PHP, то Apache передает ее интерпретатору PHP. Там и выполняется PHP-код. Результат выполнения этого кода встраивается непосредственно в HTML-файл. Интерпретатор PHP передает полученную в результате страницу обратно Apache, а сервер, в свою очередь, отсылает ее пользователю. Таким образом, в браузере не видно никакого PHP-кода, а только готовая HTML-страница (та же концепция применяется Microsoft в технологии ASP — Active Server *Pages,* «активные серверные страницы»).

Hello World! В этой книге я не смогу дать даже вводной информации о языке программирования PHP. Но приведенный далее мини-пример позволяет понять принцип действия PHP. Следующий файл после обработки PHP-интерпретатора превращается в страницу HTML, на которой указано точное время:

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0//RU">
<html><head>
  <meta http-equiv="Content-Type"
        content="text/html; charset=iso-8859-1" />
  <title>Пример PHP</title>
</head><body>

<p>Текущее время на данном сервере:
  <?php echo date(«G:i:s»); ?>
</p>
</body></html>
```

Установка. Если PHP не был установлен вместе с Apache, то вам потребуется, воспользовавшись программой управления пакетами, инсталлировать все нужные пакеты php5. Однако не так просто определить, какие пакеты действительно нужны:

подобно Apache, PHP часто разделен на несколько пакетов, в которых содержится сам язык и различные важные настройки. Чтобы начать работу с этим языком, обычно будет достаточно `php5`, `php5-common`, а также `libapache2-mod-php5`. Если после окончания установки программа управления пакетами не перезапустит Apache, сделайте это самостоятельно, чтобы веб-сервер в дальнейшем учитывал добавленный модуль PHP.

Конфигурация. Бесчисленные параметры интерпретатора PHP регулируются в файле `php.ini`. Как правило, вы можете оставить настройки, заданные по умолчанию. Место расположения данного файла зависит от дистрибутива:

- Debian, Ubuntu — `/etc/php5/apache2/php.ini`, `/etc/php5/apache2/conf.d/*.ini`;
- Fedora, Red Hat — `/etc/php.ini`, `/etc/php.d/*.ini`;
- SUSE — `/etc/php5/apache2/php.ini`.

Тестирование. Чтобы проверить, работает ли установленная копия PHP, создайте файл `phptest.php`, в котором должна быть лишь одна строка кода:

```
<?php phpinfo(); ?>
```

Скопируйте этот файл в каталог DocumentRoot и убедитесь, что Apache может читать этот файл. Хотя файлы PHP, по существу, являются сценарными, прав на чтение будет достаточно. Права на исполнение файла (биты доступа `x`) не требуются.

Теперь в браузере вы можете просмотреть страницу http://localhost/phptest.php (рис. 23.3). Она будет очень велика, и на ней будут перечислены всевозможные параметры Apache и PHP (из соображений безопасности не рекомендуется помещать такую страницу в свободном доступе в Интернете — здесь содержится масса информации о конфигурации вашего компьютера).

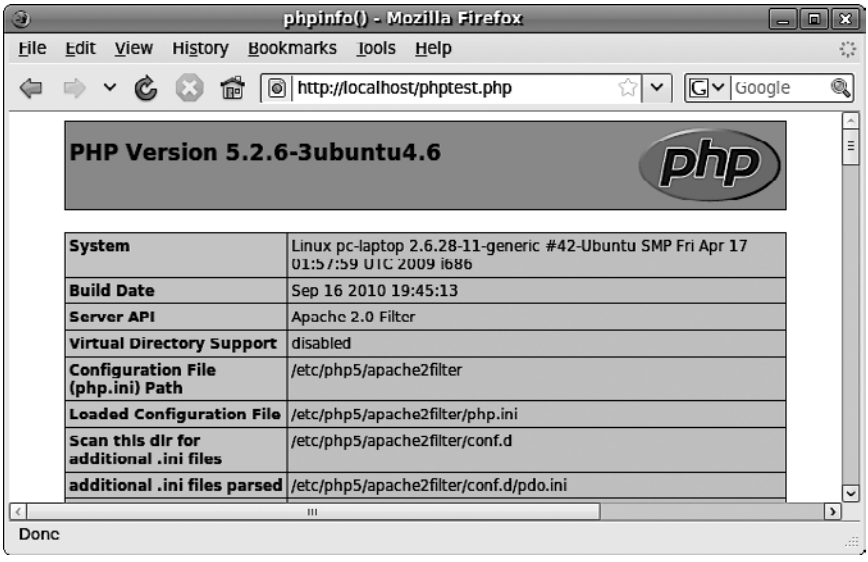


Рис. 23.3. Тестовая страница PHP

Проблемы. Если вместо тестовой страницы с кодом PHP вы получаете предложение загрузить PHP-файл, то причина ошибки, скорее всего, заключается в том, что в качестве веб-адреса было указано имя файла (например, `/srv/www/htdocs/phpinfo.php`). В таком случае файл считывается прямо из локальной системы и не обрабатывается Apache и интерпретатором PHP. Веб-адрес должен начинаться с http://!

Еще одна причина возникновения ошибки может заключаться в неверной конфигурации Apache. Не забыли ли вы перезапустить сервер Apache после того, как установили PHP или изменили конфигурационные файлы?

Если что-то не получается, посмотрите, какая информация содержится в кэше браузера, — это может дать ключ к разгадке. Вместо того чтобы снова запрашивать у Apache страницу (маловероятно, что это сработает), браузер считает ее из собственного внутреннего кэша. Ради обеспечения безопасности перезапустите программу и очистите кэш.

23.7. FTP-сервер (vsftpd)

Вместе с большинством веб-серверов используется FTP-сервер, который в зависимости от сайта может выполнять две задачи: во-первых, он обеспечивает скачивание крупных файлов, которые предоставляются на сайте; во-вторых, применяется при технической поддержке и обновлении сайта, в частности, с его помощью очень удобно закачивать файлы на сайт.

Безопасность. FTP — очень старая программа. Часто ее протокол не может работать с брандмауэрами и маскарадингом. Ситуация осложняется еще и тем, что при установлении соединения между FTP-клиентом и сервером имя пользователя и пароль передаются в незашифрованном виде. От этого у любого специалиста, неравнодушного к вопросам безопасности, волосы встают дыбом!

Разумеется, для FTP уже давно существуют безопасные аналоги. Одним из наиболее известных является описанный далее SSH-сервер с SFTP (Secure FTP), который в числе прочего обеспечивает и передачу данных. Проблема заключается, в основном, в клиентской стороне: существует сравнительно немного понятных для среднего пользователя программ, которые могут работать с SFTP, поэтому FTP, несмотря на недостаточную защищенность, все еще очень широко применяется.

Еще одна альтернатива — стандарт WebDAV, дополняющий протокол HTTP и упрощающий передачу данных в обоих направлениях. Например, Apache поддерживает WebDAV в комбинации с модулем `mod_dav`. Информацию об этом вы найдете на сайте http://httpd.apache.org/docs/2.2/mod/mod_dav.html.

Если вы не желаете отказываться от традиционного FTP-сервера, можно сконфигурировать его как «только анонимный» FTP-сервер. В таком случае при входе в систему не будет передаваться никаких важных данных. Однако FTP будет использоваться далеко не в полном объеме. Тогда вы уже не сможете применять его для обычной технической поддержки сайта.

Vsftpd. Существует множество FTP-серверов. Наиболее популярна в настоящее время программа `vsftpd`. Во всех распространенных дистрибутивах она предоставляется в виде пакета. Сочетание vsftpd означает very secure FTP daemon (очень без-

опасный демон FTP). Однако не воспринимайте слова very secure буквально, ведь даже самому безопасному FTP-серверу присущи традиционные недостатки FTP.

Запуск демона. Программу `vsftpd` можно запускать двумя способами: либо как самостоятельный демон, через систему Init-V, либо с помощью демона `xinetd`. В большинстве дистрибутивов по умолчанию задан первый вариант. Для этого в конфигурационном файле `vsftpd.conf` должна содержаться команда `listen=YES`. Чтобы запустить или остановить FTP-сервер, используйте команды, обычные для того или иного дистрибутива.

Конфигурация. Конфигурация `vsftpd` выполняется в файле `/etc/vsftpd.conf` или `/etc/vsftpd/vsftpd.conf`. Обычно по умолчанию при анонимном FTP-доступе применяется режим «только для чтения». Иначе говоря, клиенты FTP могут скачивать файлы, а закачивать не могут. Если вам нужно, чтобы на FTP можно было входить не только анонимно, но и под определенным логином, то установите для `locale_enable` значение `YES`. Если при использовании такой формы FTP вы также хотите разрешить закачку файлов, установите для `write_enable` значение `YES`. Если в файле `vsftpd.conf` содержится строка `tcp_wrappers=Yes`, то программа `vsftpd` интерпретирует файлы `/etc/hosts.allow` и `/etc/hosts.deny` (см. раздел 26.2). В следующем примере обобщаются важнейшие настройки `vsftpd.conf`:

```
# /etc/vsftpd.conf или /etc/vsftpd/vsftpd.conf
...
local_enable=YES / NO        # Допуск FTP-логина
write_enable=YES / NO        # Разрешение на закачку файлов
...
anonymous_enable=YES / NO    # Разрешение анонимного доступа по FTP
anon_upload_enable=YES / NO  # Разрешение закачки файлов,
                             # в том числе при анонимном доступе по FTP
...
listen=YES / NO              # Запуск в качестве демона Init-V-Damon (YES)
                             # или xinetd (NO)
tcp_wrapper=YES / NO         # Интерпретация hosts.allow и hosts.deny
```

Тестирование FTP. Теперь FTP должен заработать. Выполните на сервере команду `ftp localhost`, чтобы проверить, правильно ли был запущен сервер. При этом учитывайте, что `root`, как правило, не должен указывать FTP-логин.

Анонимный доступ по FTP

Если в `vsftpd.conf` допускается анонимный доступ по FTP, `vsftpd` принимает в качестве логина имена `anonymous` и `ftp` в комбинации с любым паролем. Обычно в качестве пароля указывается адрес электронной почты, но `vsftpd` этого не контролирует.

После входа в систему FTP-клиент получает доступ к файлам домашнего каталога пользователя Linux **ftp**. Местоположение этого каталога указывается в `/etc/passwd`:

- Debian, Ubuntu — `/home/ftp`;
- Fedora, Red Hat — `/var/ftp/`;
- SUSE — `/srv/ftp/`.

Закачка файлов через анонимный FTP. Если вы разрешаете загружать в систему файлы через анонимный FTP, обратите внимание на то, что в каталоге с данными FTP имеется лишь один каталог, предоставляющий право на внесение изменений, например `/var/ftp/upload` в Fedora или Red Hat. Этот каталог должен принадлежать пользователю FTP и из соображений безопасности не давать прав на чтение файлов:

```
root# mkdir /var/ftp/upload
root# chown ftp upload
root# chmod 730 upload
```

Таким образом, любому пользователю разрешается закачивать файлы в систему, а также отправлять системному администратору электронное сообщение с разъяснениями, для чего нужен закачанный файл. Остальные пользователи не могут ни просмотреть, ни скачать этот файл из каталога upload. Если не принять таких мер безопасности, то ваш каталог для закачек FTP можно будет использовать для нелегального обмена файлами.

FTP для администратора и других особых категорий пользователей

По соображениям, связанным с безопасностью, администратор (root) и некоторые другие особые пользователи, в частности daemon, lp или nobody, не могут применять FTP. Конфигурация, при которой реализуется такой запрет, отличается от дистрибутива к дистрибутиву.

В Fedora и Red Hat защита логина происходит двумя способами одновременно. С одной стороны, для контроля логина `vsftpd` обращается к PAM (подключаемым модулям аутентификации). PAM интерпретируют файл `/etc/pam.d/vsftpd`, указывающий на файл `etc/vsftpd/ftpusers`. В последнем содержится список всех логинов, которые *не могут* использовать FTP.

С другой стороны, в FTP применяется и внутрисистемный контроль логинов — все пользователи, названные в `/etc/vsftpd.user_list`, блокируются. Такой контроль логинов активизируется в `vsftpd.conf` с помощью `userlist_enable=YES` и `userlist_deny=YES` (действует по умолчанию).

В Debian, SUSE и Ubuntu `vsftpd` обращается к логину также через PAM. Здесь `/etc/pam.d/vsftpd` всегда указывает на `/etc/ftpusers`. В этом файле содержится список всех логинов, с которых нельзя использовать FTP.

24 MySQL

Уже много лет MySQL является самой популярной системой баз данных в мире свободного ПО. Даже если вы не планируете самостоятельно управлять какими-либо базами данных, существует множество веб-приложений, для работы с которыми требуется MySQL-сервер. Это и различные «Вики», дискуссионные форумы наподобие phpBB, системы управления содержимым, например TYPO3, а также системы управления ошибками, например Bugzilla.

В этой главе будет рассказано, как установить на компьютере MySQL-сервер и как выполнять самые важные задачи по его администрированию. При этом я не буду затрагивать таких тем, как дизайн баз данных, применение SQL для опроса или для управления данными. Я также опущу вопросы разработки приложений для работы с базами данных — на все это в книге просто не хватит места.

24.1. MySQL

По сравнению с коммерческими базами данных, например Oracle или IBM DB/2, в MySQL предлагается меньше функций, но для решения большинства практических задач набора ее функций вполне достаточно. Информация о MySQL, а также очень подробный учебник имеется по следующему адресу: http://dev.mysql.com.

Лицензия. MySQL, а также ее драйверы для работы с различными языками программирования соответствуют лицензии GPL. Использование MySQL в рамках отдельно взятой фирмы (без передачи или коммерческой реализации) является бесплатным. Но учитывайте, что для передачи коммерческих проектов *с закрытым кодом*, не соответствующих GPL и построенных на основе MySQL, требуется коммерческая лицензия MySQL-сервера. Вопрос лицензирования MySQL подробно рассмотрен на сайте http://www.mysql.com/about/legal/.

В январе 2010 года компания Oracle приобрела Sun и теперь является собственником MySQL. Смена собственника не обошлась без шероховатостей. В частности, сообщество свободных разработчиков критикует тот факт, что при поставке обновлений безопасности Oracle отдает предпочтение корпоративным клиентам.

После этого появилось множество проектов-ответвлений. Наряду с официальным MySQL-сервером, который по-прежнему бесплатно доступен на сайте MySQL, вашему вниманию предлагается сервер MariaDB (http://askmonty.org/) — проект Монти Видениуса, отца-основателя MySQL, а также сервер Percona

(http://www.percona.com), авторства фирмы Percona, занятой поддержкой MySQL. Как правило, перед использованием эти пакеты требуется установить вручную. Только в openSUSE пакеты MariaDB предоставляются сразу, но по умолчанию даже в openSUSE применяется сервер MySQL.

Установка и обеспечение безопасности

Во всех распространенных дистрибутивах есть пакеты с MySQL. Обратите внимание, что сам сервер базы данных, его библиотеки, инструменты администрирования и другое устанавливаются в различных пакетах. Далее указано, какие пакеты вам могут понадобиться:

- Debian, Ubuntu — `mysql-client`, `mysql-common`, `mysql-server`;
- Fedora, RHEL — `mysql`, `mysql-server`;
- SUSE — `mysql-community-server`, `mysql-community-server-client`.

В Fedora целесообразно устанавливать не отдельные пакеты, а заранее составленную группу пакетов MySQL:

```
root# yum groupinstall mysql
```

Запуск/установка. MySQL — это демон, который в отдельных дистрибутивах нужно специально запускать. Команда, используемая для этого, в различных дистрибутивах называется по-разному. В обобщенном виде эта информация дана в разделе 8.5. В зависимости от дистрибутива сценарий Init-V называется `mysql` или `mysqld`.

Файлы базы данных сервера MySQL обычно сохраняются в каталоге `/var/lib/mysql`. Файлы регистрации чаще всего находятся в каталогах `/var/log/syslog` (Debian, Ubuntu), `/var/lib/mysql/`*хост-имя* (SUSE) или `/var/log/mysql*`.

Конфигурация. Конфигурация сервера MySQL выполняется в файле `/etc/my.cnf` или `/etc/mysql/my.cnf` (Debian, Ubuntu). Этот файл уже подготовлен для решения повседневных задач. Ради экономии места я не буду описывать здесь все ключевые слова данного файла, но остановлюсь на некоторых деталях, важных с точки зрения безопасности.

Как правило, конфигурация разбивается на несколько фрагментов указанием вида `[`*имя*`]`. Далее мы рассмотрим только раздел `[mysqld]`, касающийся самого сервера. Требования, предъявляемые к этому разделу, вступают в силу только после перезапуска сервера MySQL. Остальные разделы относятся к конфигурации различных клиентских программ.

- `old_passwords=1` — благодаря данной настройке MySQL сохраняет внутренние пароли так, как это делалось в версиях MySQL 4.0 и ниже. В некоторых дистрибутивах данная настройка все еще используется по умолчанию, позволяя обойти проблемы совместимости, возникающие при работе со старыми экземплярами MySQL. Однако настройка сильно снижает защищенность системы доступа к MySQL.

  Если вы собираетесь работать и без оглядки на старые версии MySQL, по возможности следует использовать вариант `old_passwords=0` или просто удалить эту

команду из `my.cnf` *до того*, как настраивать какие-либо пароли! Чтобы измененные настройки вступили в силу, перезапустите сервер MySQL. Подробная базовая информация о старой и новой системах аутентификации дается в учебнике по MySQL по адресу http://dev.mysql.com/doc/refman/5.1/en/password-hashing.html.

- `bind-address=127.0.0.1` — при такой настройке сетевое соединение с сервером MySQL можно установить лишь с локального компьютера (но ни с какого-либо другого компьютера из тех, что находятся в локальной сети или в Интернете). Если MySQL должна использоваться только локальными программами (например, PHP), то данная настройка помогает повысить безопасность. В Debian и Ubuntu настройка действует по умолчанию, а в остальных дистрибутивах ее нужно добавить, если это возможно.
- `skip-networking` — закрывает любой доступ из сети к серверу MySQL, даже сетевые соединения, устанавливаемые с локального компьютера. Таким образом, это еще более ограничительная настройка, чем `bind-address=127.0.0.1`. В данном случае соединение с сервером могут устанавливать только локальные программы, обменивающиеся информацией с сервером MySQL через так называемый сокет-файл, например, программы, написанные на PHP и на C. Программы, которые обмениваются с MySQL-сервером информацией по протоколу TCP/IP (например, все программы Java), не смогут работать с MySQL-сервером! При возникновении сомнений используйте `bind-address=127.0.0.1`.

Пароль администратора

В Debian и Ubuntu в ходе установки сервера MySQL вам понадобится указать пароль администратора. Как и в Linux, в MySQL такой пароль имеет особое значение и предоставляет неограниченные права администрирования.

Кроме того, в Debian и Ubuntu создается учетная запись `debian-sys-main`, защищенная паролем, который генерируется случайным образом и сохраняется в файле `/etc/mysql/debian.cnf`, где в виде незашифрованного текста он доступен только администратору Linux. Сценарий `/etc/mysql/debian-start`, необходимый для запуска MySQL, обращается к этой учетной записи, поэтому учетную запись `debiansys-main` ни в коем случае нельзя отключать! При изменении пароля нужно соответствующим образом обновить и `debian.cnf`.

В Fedora, Red Hat и SUSE такая дополнительная защита MySQL не предусмотрена, поэтому после завершения установки любой пользователь, применив логин `root`, может без пароля установить соединение с MySQL. Для этого используются следующие команды:

```
user$  mysql -u root
mysql> UPDATE mysql.user SET password=PASSWORD('xxx') WHERE user='root';
mysql> FLUSH PRIVILEGES;
mysql> exit
```

Если в дальнейшем вы планируете выполнять команды SQL с помощью `mysql`, войти в систему нужно так:

```
user$ mysql -u root -p
Enter password: ******
```

Логины и пароли, используемые в Linux и MySQL, никак не связаны друг с другом. Из соображений безопасности ни в коем случае не применяйте для одного и того же пользователя Linux и MySQL одинаковый пароль. Часто пароли MySQL сохраняются в программном коде, поэтому защитить их гораздо сложнее, чем обычные пароли Linux.

Анонимные пользователи MySQL

Теперь пользователь MySQL `root` защищен паролем. Но в некоторых дистрибутивах с базой данных могут работать и анонимные пользователи. Это означает, что на сервер MySQL может войти любой пользователь с любым логином. Хотя права анонимных пользователей и сильно ограничены, такой пользователь может быть опасен и лучше не допускать его в систему.

Чтобы узнать, есть ли в системе анонимные пользователи, выполните команду `mysql`. У анонимных пользователей в результате выполнения команды `SELECT` столбец `user` не будет содержать никакой информации. С помощью команды `DELETE` таких пользователей можно удалить. Команда `FLUSH PRIVILEGES` сразу же активизирует подобное изменение базы данных.

```
user$ mysql -u root -p
Enter password: *******
mysql> SELECT user, host, password FROM mysql.user;
+------+---------------+-------------------------------------------+
| user | host          | password                                  |
+------+---------------+-------------------------------------------+
| root | localhost     | *AFCF054603403E6863B8DCFC1BEAC269746E8720  |
| root | <hostname>    | *AFCF054603403E6863B8DCFC1BEAC269746E8720  |
| root | 127.0.0.1     | *AFCF054603403E6863B8DCFC1BEAC269746E8720  |
| root | ::1           | *AFCF054603403E6863B8DCFC1BEAC269746E8720  |
|      | localhost     |                                           |
|      | <hostname>    |                                           |
+------+---------------+-------------------------------------------+
mysql> DELETE FROM mysql.user WHERE user='';
mysql> FLUSH PRIVILEGES;
mysql> exit
```

Первые тесты

Создание новой базы данных и новых учетных записей пользователей. Чтобы протестировать MySQL, вы должны знать язык SQL, предназначенный для работы с базами данных. В данном случае я предполагаю, что вы действительно его знаете. Цель команды, показанной ниже, — создать новую базу данных `mydatabase` и новую учетную запись пользователя `newuser`, который будет иметь право доступа к этой базе. Для решения таких задач лучше всего использовать программу `mysql`. Она нужна примерно для того же, что и shell в Linux: принимает команды MySQL, переадресовывает их на MySQL-сервер и, наконец, отображает результат. При этом все команды MySQL должны оканчиваться точкой с запятой.

```
user$ mysql -u root -p
Enter password: ******
```

```
...
mysql> CREATE DATABASE mydatabase;
mysql> GRANT ALL ON mydatabase.* TO newuser@localhost
       IDENTIFIED BY 'xxxxxxxxx';
mysql> exit
```

Пользователь `newuser` может проводить с новоиспеченной базой данных любые другие тесты. Как и в Linux, в MySQL лучше как можно меньше работать под учетной записью `root`.

Создание таблиц и заполнение их данными. С помощью следующих команд `newuser` создает новую таблицу (`CREATE TABLE`), вставляет в нее определенные наборы данных (`INSERT`) и, наконец, просматривает все наборы данных (`SELECT`).

Особое значение в данной таблице имеет столбец `id`: сервер MySQL самостоятельно вставляет туда каждый новый набор данных, присваивая ему уникальный номер, который потом используется для идентификации этого набора данных.

```
user$ mysql -u newuser -p
Enter password: ******
mysql> USE mydatabase;
mysql> CREATE TABLE mytable (
         id INT NOT NULL AUTO_INCREMENT,
         txt VARCHAR(100),
         n INT,
         PRIMARY KEY(id));
mysql> INSERT INTO mytable (txt, n) VALUES('abc', 123);
mysql> INSERT INTO mytable (txt, n) VALUES('efgsd', -4);
mysql> INSERT INTO mytable (txt, n) VALUES(NULL, 0);
mysql> SELECT * FROM mytable;
+----+-------+------+
| id | txt   | n    |
+----+-------+------+
| 1 | abc    | 123  |
| 2 | efgsd  |  -4  |
| 3 | NULL   |   0  |
+----+-------+------+
mysql> exit
```

24.2. Администрирование MySQL

Существует масса инструментов для администрирования MySQL. По умолчанию предоставляются команды `mysql`, `mysqldump`, а также некоторые другие текстовые инструменты. При желании вы можете установить и другие программы, для части из которых, правда, требуется графическое окружение Рабочего стола. В этом разделе мы кратко рассмотрим важнейшие инструменты.

Здесь не будут описаны команды `mysqldump` и `mylvmbackup`, предназначенные для резервного копирования. Мы отдельно обсудим тему резервного копирования в MySQL в разделе 24.3.

Mysql. Программа `mysql` представляет собой не сервер MySQL (сервер называется `mysqld`), а клиент командной строки. Он позволяет установить связь с MySQL-сервером, а после этого выполнять SQL-команды. Пока работа касается только команд `SELECT`, программа отображает результаты запросов в текстовом режиме.

При запуске mysql вы, как правило, с помощью `-u` указываете имя пользователя MySQL. Благодаря параметру `-p` после запуска вы должны указывать соответствующий пароль. Если сервер MySQL работает не на локальном компьютере, то хостимя указывается с помощью `-h`. Можно также указать и стандартную базу данных, которая будет использоваться как основной источник данных для всех остальных SQL-команд.

```
user$ mysql -u root -p mojabazadannyh
Enter password: *******
```

После этого можно вручную задавать SQL-команды, завершаемые точкой с запятой. Специфичная для `mysql` команда `status`, которая срабатывает и без точки с запятой, отображает информацию по актуальному соединению, а также основные характеристики MySQL-сервера. Сочетание клавиш Ctrl+D завершает программу.

```
mysql> status
mysql Ver 14.14 Distrib 5.5.22, for debian-linux-gnu (x86_64) using readline 6.2
...
Connection id:          48097
Server version:         5.5.22-0ubuntu1 (Ubuntu)
Protocol version:       10
Connection:             Localhost via UNIX socket
Server characterset:    latin1
Db characterset:        latin1
Client characterset:    utf8
Conn. characterset:     utf8
UNIX socket:            /var/run/mysqld/mysqld.sock
Uptime:                 7 days 13 hours 31 min 33 sec
...
```

Кодировка UTF-8. Как видно из приведенного выше примера, при обмене информацией между `mysql` и сервером MySQL обычно применяется кодировка «латиница-1» (`latin1`). В результате международные символы в текстовых консолях отображаются неверно, поскольку там по умолчанию действует кодировка UTF-8. Справиться с этой проблемой помогает следующая команда SQL:

```
mysql> SET NAMES utf8;
```

«Более интеллектуальные» клиентские программы сами отвечают за правильную настройку кодировки. MySQL также способна разобраться с любыми кодировками. Нужно обращать внимание лишь на то, чтобы при создании новых таблиц вы выбирали правильную кодировку для текстовых столбцов.

Mysql в пакетном режиме. Для целей администрирования и, в частности, для слаженной работы с механизмом резервного копирования, mysql также может обрабатывать SQL-команды из файла `*.sql`.

```
user$ mysql -u root -p mojabazadannyh < kommandos.sql
Enter password: *******
```

Mysqladmin

`Mysqladmin` позволяет выполнять различные задачи по администрированию в форме команд. Для всех команд `mysqladmin` существуют равнозначные команды SQL (например, `CREATE DATABASE`). Преимущество `mysqladmin` заключается в том, что она позволяет автоматизировать повторяющиеся задачи с помощью сценариев.

Как и при работе с `mysql`, с помощью параметра `-u` вы указываете имя пользователя, а используя `-h` — хост-имя (по умолчанию это `localhost`). Задавая параметр `-p` без других указаний, мы задействуем интерактивное запрашивание пароля. Чтобы избежать введения пароля, укажите `-p`*`password`* (без пробелов после `-p`). Тем не менее, у этого удобства есть и отрицательная черта: пароль пересылается в виде обычного текста. Альтернативный вариант заключается в том, чтобы сохранять данные о входе в систему в файле пароля либо задействовать пользователя MySQL `debian-sys-maint`, по умолчанию задаваемого в Debian и Ubuntu. Файл с паролем этого пользователя `/etc/mysql/debian.cnf` доступен для чтения только администратору Linux. В таком случае вызов `mysqladmin` будет выглядеть следующим образом:

```
root# mysqladmin --defaults-file=/etc/mysql/debian.cnf kommando ...
```

Чтобы просмотреть команды, доступные при работе с `mysqladmin`, введите запрос `mysqladmin --help`. В следующих строках приведено несколько примеров: первая команда создает новую базу данных, вторая выдает список всех статусных переменных MySQL, третья выводит список всех активных MySQL-потоков (соединений).

```
user$ mysqladmin -u root -p create новаябазаданных
Enter password: *******
user$ mysqladmin -u root -p extended-status
...
user$ mysqladmin -u root -p processlist
...
```

MySQL Workbench

MySQL Workbench — великолепная программа для администрирования MySQL, снабженная графическим пользовательским интерфейсом (рис. 24.1). Она позволяет отслеживать работу MySQL-сервера, изменять его параметры, настраивать права пользователей, создавать новые базы данных, считывать, изменять и защищать имеющиеся базы данных и т. д.

В большинстве дистрибутивов MySQL Workbench предоставляется в отдельных пакетах. Печально знаменитым исключением является RHEL. Чтобы применять MySQL Workbench в дистрибутивах, основанных на Red Hat, нужно скачать пакеты непосредственно с сайта MySQL: http://dev.mysql.com/downloads/workbench.

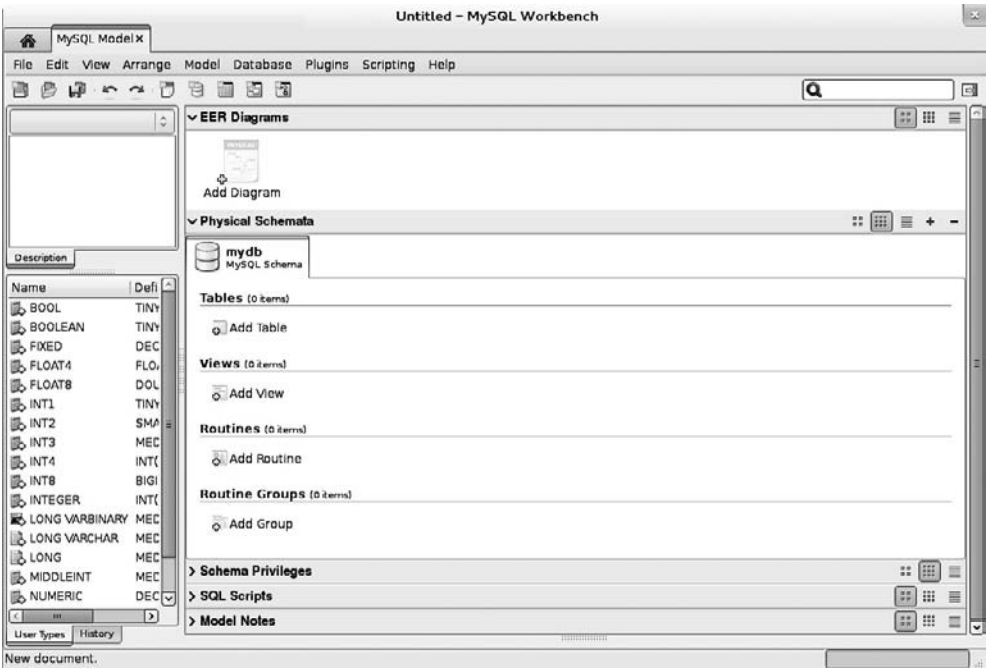


Рис. 24.1. MySQL Workbench

PhpMyAdmin

Вероятно, самой известной программой для администрирования MySQL является phpMyAdmin. Ее основное преимущество заключается в том, что программу можно использовать с помощью браузера откуда угодно. Сам код phpMyAdmin выполняется непосредственно на сервере. Этот механизм работает и в тех случаях, когда из соображений безопасности сервер MySQL конфигурируется без разрешения устанавливать какие-либо соединения через сеть.

Во многих дистрибутивах есть готовые пакеты с phpMyAdmin. Если в вашем дистрибутиве такой пакет отсутствует, скачайте phpMyAdmin с сайта http://www.phpmyadmin.net/home_page/ и установите PHP-файлы в каталог, к которому открыт доступ для веб-сервера Apache.

Debian, Ubuntu. При конфигурировании phpMyAdmin для работы с Debian и Ubuntu учитывайте, что все конфигурационные файлы находятся в каталоге `/etc/phpmyadmin`. Если вы используете в качестве веб-сервера Apache, то `apache.conf` отвечает за отображение фактического установочного каталога `/usr/share/phpmyadmin` на веб-каталог. Это делается с помощью псевдонима. Настройки безопасности, заданные в `apache.conf`, затрагивают, как правило, лишь установочный сценарий phpMyAdmin, который вам обычно вообще не понадобится.

`Apache.conf` интегрируется в конфигурацию Apache с помощью символьной ссылки `/etc/apache2/conf.d`. Если вы хотите обмениваться информацией с phpMyAdmin по протоколу HTTPS, то удалите эту ссылку и вставьте в ваш конфигурационный файл HTTPS команду `Include` для `apache.conf`:

```
<VirtualHost _default_:443>
    ...
    Include /etc/phpmyadmin/apache.conf
</VirtualHost>
```

Конфигурация. За детали конфигурации phpMyAdmin отвечает файл `config.inc.php`. Здесь нужно вносить изменения лишь в том случае, когда вы планируете пользоваться теми или иными специальными функциями phpMyAdmin. Но для работы с такими функциями обязательно требуется создать новую базу данных MySQL, в которой phpMyAdmin сможет сохранять свою информацию. Более подробная информация приведена по адресу http://www.phpmyadmin.net/documentation/#linked-tables.

Защита phpMyAdmin. phpMyAdmin — излюбленная злоумышленниками точка проникновения в систему. В Интернете есть масса автоматизированных инструментов, отыскивающих на сервере плохо защищенные или устаревшие версии phpMyAdmin. Если пароль администратора к MySQL отсутствует либо его легко отгадать, злоумышленник может заполучить на MySQL-сервере права администратора! Чтобы этого не случилось, примите следующие меры предосторожности:

- защитите ваш экземпляр базы данных MySQL, прежде чем устанавливать phpMyAdmin;
- задайте для каталога с phpMyAdmin псевдоним, который не так легко отгадать. http://mysite.ru/pMa1 — надежнее, чем http://mysite.ru/phpmyadmin;
- по возможности применяйте при работе с каталогом phpMyAdmin только безопасный протокол HTTPS, а не HTTP;
- защитите каталог с phpMyAdmin на уровне Apache паролем либо, например, с помощью файла `.htaccess` (см. раздел 23.2).

Chive

Программа phpMyAdmin широко распространена, но не пользуется бесспорной популярностью среди пользователей. Работа с ней довольно запутанна, различные дополнительные функции требуют сложной конфигурации (в Debian и Ubuntu такая конфигурация может быть выполнена автоматически в ходе установки пакета phpMyAdmin). Кроме того, репутация phpMyAdmin пострадала из-за многочисленных имевших место проблем с безопасностью.

Новый проект Chive (http://www.chive-project.com/) призван заменить функционал phpMyAdmin с помощью более современных и удобных приемов (рис. 24.2). Chive не может соперничать с phpMyAdmin в разнообразии функций, но во многих случаях этой программы вполне достаточно. В области удобства использования Chive уже явственно превосходит phpMyAdmin. Например, здесь предлагается строковый редактор для изменения отдельных полей с данными.

Установка. Пока отсутствуют дистрибутивы, в которых Chive предлагался бы в виде самостоятельного пакета. В Ubuntu, правда существует Personal Package Archive (PPA, персональный архив пакетов), в котором имеются пакеты Chive. Но, как правило, программу приходится устанавливать вручную. Это несложно. Вы просто скачиваете с сайта Chive TAR-архив и распаковываете его в каталоге, доступном для Apache.

Рис. 24.2. Администрирование MySQL с помощью Chive

После этого еще потребуется изменить пользователя файлов, чтобы Apache получил к ним доступ. В Debian и Ubuntu необходимые команды выглядят так:

```
root# cd /var/www/
root# wget -O – http://www.chive-project.com/Download/Redirect|tar -xzp
root# chown -R www-data chive
```

Далее Chive запускается в браузер по следующему адресу:

```
http://mysite.ru/chive/index.php
```

Защита. Сложно сказать, насколько хороша программа Chive с точки зрения безопасности — проект пока слишком новый. Но в принципе при работе с Chive действуют те же правила, что и с phpMyAdmin (см. выше). Защитите каталог Chive паролем, по возможности используйте HTTPS и избегайте таких паролей к MySQL, которые легко отгадать!

24.3. Резервное копирование

Если вы и не собираетесь специализироваться на администрировании MySQL, вы, тем не менее, должны знать, как создать исправную резервную копию базы данных MySQL. Для этого существует больше вариантов и стилей, чем можно себе представить. В этом разделе мы поговорим о командах `mysqldump` и `mylvmbackup`. Если вам требуются не только такие резервные копии, которые делаются с регулярными интервалами, но и непрерывное резервное копирование, активизируйте (двоичное) логирование. В таком случае любое изменение, осуществляемое в базе данных, будет фиксироваться в файле логов. Файлы логов можно также использовать в качестве основы для репликации базы данных на втором сервере.

Типы таблиц. Многообразие методов резервного копирования при работе с MySQL связано, в частности, с тем, что в MySQL известны различные типы таб-

лиц. При создании таблицы разработчик базы данных (либо программа) может выбирать из таких типов таблиц. Иногда в одной базе данных применяются таблицы различных типов. В настоящее время двумя важнейшими типами таблиц являются MyISAM и InnoDB. В зависимости от типа таблицы вы не просто располагаете разными наборами дополнительных функций (трансакции, правила внешних ключей, полнотекстовый поиск). Варьируются также базовый формат данных и оптимальный метод резервного копирования.

Если вы не знаете, какими базами данных управляет ваш MySQL-сервер и к какому типу относятся таблицы этой базы данных, выполните в клиенте командной строки `mysql` следующую SQL-команду. В полученном результате будут перечислены все таблицы. В столбце `table_schema` при этом указывается имя базы данных, в столбце `engine` — тип таблицы.

```
mysql>SELECT table_schema, table_name, engine
      FROM information_schema.tables
      WHERE table_type='BASE TABLE'
      ORDER BY table_schema, table_name;

table_schema table_name      engine
-----------------------------------
mylibrary    authors         InnoDB
mylibrary    categories      InnoDB
mylibrary    counters        MyISAM
mylibrary    fulltitles      MyISAM
mylibrary    languages       InnoDB
...
```

Mysqldump

Команда `mysqldump`, входящая в стандартный набор программ для работы с MySQL, создает резервную копию базы данных MySQL в форме SQL-команд. Результирующий файл позже можно внедрить в уже имеющийся файл с помощью `mysql`. В принципе синтаксис таков:

```
user$ mysqldump -u root -p [параметры] имябазыданных > backup.sql
```

Детали резервного копирования регулируются с помощью многочисленных параметров (см. также `mysqldump --help`). Оптимальная комбинация параметров зависит, в частности, от того, в каком формате записаны таблицы вашей базы данных MySQL (MyISAM или InnoDB).

Резервное копирование таблиц MyISAM. При копировании таблиц MyISAM достаточно параметра `--lock-all-tables`. Благодаря этому параметру в начале резервного копирования `mysqldump` блокирует *все* таблицы с помощью команды `LOCK` и высвобождает лишь по завершении резервного копирования. Но по умолчанию `mysqldump` блокирует таблицу за таблицей (то есть в данный момент блокируется только та таблица, которая в текущий момент обрабатывается). В результате может случиться так, что во время резервного копирования отдельные таблицы изменятся и что связи между таблицами станут неверными.

```
user$ mysqldump -u root -p --lock-all-tables myisam-db > backup.sql
```

Резервное копирование таблиц InnoDB. Гораздо больше параметров требуется для резервного копирования базы данных с таблицами InnoDB. С помощью `--skip-opt` деактивируются некоторые стандартные параметры, специально предназначенные для работы с MyISAM. Благодаря параметру `--single-transaction` вся операция резервного копирования производится за одну трансакцию. Таким образом, во время резервного копирования становится невозможно изменять данные и, следовательно, между таблицами могут возникать неверные связи. При использовании параметра `--disable-keys` при последующем считывании данных временно деактивируется обновление индекса. Только по завершении резервного копирования индекс полностью воссоздается заново, работа идет значительно быстрее. Благодаря `--create-options`, команда `mysqldump` использует при выводе MySQL-специфичные параметры команды `CREATE-TABLE`.

При применении параметра `--quick`, `mysqldump` данные считываются с сервера множество за множеством, а не все сразу. Таким образом работа с большими таблицами получается более эффективной. На основе `--extended-insert` команда `mysqldump` создает команды `INSERT`, вставляющие несколько множеств данных сразу. В результате, во-первых, несколько уменьшается объем файла резервной копии и, во-вторых, повторная интеграция данных впоследствии ускоряется. Благодаря параметру `--add-drop-table` команда `mysqldump` предваряет каждую команду `CREATE-TABLE` командой `DROP-TABLE`. Так удается избежать ошибок, если в базе данных уже существуют какие-то таблицы. Например, такие таблицы могли образоваться в результате не полностью законченного резервного копирования.

```
user$ mysqldump -u root -p --skip-opt --single-transaction \
      --disable-keys --create-options --quick \
      --extended-insert --add-drop-table inno-db > backup.sql
```

Остается поговорить о том, как лучше всего организовать работу, если в базе данных содержатся таблицы как в формате MyISAM, так и InnoDB. В таком случае `mysqldump` просто используется без дополнительных параметров. Но если база данных применяется во время резервного копирования, то может случиться так, что в ходе копирования таблицы изменятся. Если так произойдет, то база данных станет дефектной, то есть связанные друг с другом таблицы будут содержать ссылки на уже не существующие множества данных. С помощью `mysqldump` эта проблема не решается. Лучше всего воспользоваться другим методом резервного копирования (см. описание `mylvmbackup` далее).

Если вы создаете резервную копию *всех* баз данных (а не конкретной базы данных), то укажите параметр `--all-databases`.

Хранимые процедуры и триггеры. Обратите внимание, что по умолчанию `mysqldump` не сохраняет ни хранимых процедур (stored procedure), ни триггеров. Если вы хотите сохранять такую информацию, то дополнительно укажите параметры `--routines` и `--triggers`.

Восстановление базы данных. Чтобы восстановить базу данных из резервной копии, сначала создайте эту базу данных (если она еще не существует). Затем передайте `mysql` файл резервной копии. При этом параметр `-default-character-set` гарантирует, что последовательности символов, сохраненные в кодировке UTF-8, также будут считываться именно в этом формате.

```
user$ mysqladmin create dbname
user$ mysql -u root -p --default-character-set=utf8 dbname < backup.sql
```

Заархивированные резервные копии. Файлы резервных копий, создаваемые с помощью `mysqldump`, очень велики из-за своего текстового формата. С этой проблемой проще всего справиться с помощью архивации баз данных, и разархивировать их только перед тем, как они снова будут использоваться. Соответствующие команды выглядят так:

```
user$ mysqldump [параметры] dbname | gzip -c > backup.sql.gz
user$ gunzip -c backup.sql.gz | mysql [параметры] dbname
```

Но при сжатии данных с помощью `gzip` тратится множество ресурсов процессора. Если вы хотите сэкономить процессорное время и можете позволить себе хранить сравнительно объемные резервные копии, то рекомендуется использовать команду сжатия `lzop`, находящуюся в одноименном пакете:

```
user$ mysqldump [параметры] dbname | lzip -c > backup.sql.lzo
user$ lzop -c -d backup.sql.lzo | mysql [параметры] dbname
```

Mylvmbackup

Команда `mysqldump` отлично подходит для работы с небольшими базами данных. В любом случае в ходе резервного копирования таблицы доступны лишь для ограниченного применения. В случае с таблицами MyISAM это объясняется блокировкой, в случае с InnoDB — длительностью трансакции. В системах баз данных, которые должны круглосуточно работать без перерывов, все время растущий размер баз данных становится довольно серьезной проблемой.

Оптимальным решением подобной проблемы является «горячее» резервное копирование, которое можно с легкостью осуществлять прямо на ходу — во время работы базы данных. Для таблиц InnoDB существует соответствующая программа для резервного копирования, которая, однако, доступна только по платной корпоративной лицензии MySQL.

Если вы не хотите приобретать такую программу за деньги, то описанная здесь команда `mylvmbackup` является хорошей альтернативой. `Mylvmbackup` минимизирует время, в течение которого база данных остается заблокированной. Как правило, этот период не превышает нескольких секунд (независимо от размера самой базы данных!). В любом случае предполагается, что все базы данных должны находиться в разделе логического тома (Logical Volume, LVM).

Во многих дистрибутивах сценарий `mylvmbackup` на языке Perl можно устанавливать в виде пакета. Если в вашем дистрибутиве такой пакет не предоставляется, то скачайте сценарий с сайта http://www.lenzg.net/mylvmbackup/.

Основы. Далее предполагается, что вы уже умеете работать с менеджером логических томов (см. раздел 14.15). Здесь мы вспомним о том, что логические тома позволяют создавать так называемые мгновенные снимки такого тома. В результате вы можете фактически «заморозить» содержимое определенного логического тома и в таком состоянии добавить эту информацию в файловую систему как новый логический том.

Исходный логический том можно изменять дальше. В любом случае работать с мгновенным снимком логического тома вы сможете лишь при условии, что в системе найдется достаточно места для копий всех измененных блоков с данными. Эти блоки данных копируются в буфер мгновенного снимка. Размер данного буфера вы задаете сами на этапе создания снимка. С внутрисистемной точки зрения LVM копирует каждый блок с данными перед внесением изменений. Таким образом, мгновенный снимок содержит старое состояние, а исходный логический том применяет измененный блок данных.

`Mylvmbackup` использует мгновенный снимок LVM, чтобы защитить все файлы базы данных из каталога `/var/lib/mysql`. Подробно процесс выглядит так.

- В результате выполнения команды SQL `FLUSH TABLES WITH READ LOCK` вся информация MyISAM сохраняется на жестком диске, после чего блокируются дальнейшие изменения, которые могли бы быть осуществлены другими клиентами. Иногда на выполнение этой команды уходит достаточно много времени, поскольку MySQL-сервер должен завершить все другие команды LOCK, задействованные ранее.
- `lvcreate -s` создает мгновенный снимок LVM и добавляет этот том в файловую систему. Как минимум мгновенный снимок должен содержать каталог `/var/lib/mysql`.
- `SHOW MASTER STATUS` узнает, какой двоичный файл логирования активен в настоящий момент и где он находится. Эта информация записывается в файл резервного копирования `backup-pos/*_mysql.pos`. Такие данные будут важны в случае, если резервная копия позже будет использоваться при создании системы репликации либо если должны применяться инкрементные резервные копии из файлов логов.
- `UNLOCK TABLES` вновь высвобождает таблицы MyISAM. После этого сервер MySQL опять доступен вам в полном объеме. (Таблицы InnoDB вообще не блокируются, поэтому их не нужно высвобождать.)
- Теперь мгновенный снимок логического тома LVM подключается к файловой системе и резервные копии всех файлов из каталога `/var/lib/mysql` записываются в TAR-архив. В больших базах данных на это определенно требуется значительное количество времени. Данное время зависит и от того, насколько сильно жесткий диск загружен выполнением других операций.
- Кроме того, в TAR-архив добавляется копия `/etc/mysql/my.cnf`.
- Наконец, мгновенный снимок LVM снова извлекается из файловой системы, после чего удаляется.

Может возникнуть вопрос: а почему мы ничего не делаем для синхронизации файлов InnoDB. Этого не приходится выполнять, поскольку трансакции при работе с таблицами InnoDB выполняются в соответствии с требованиями. В любой момент драйвер InnoDB может полностью восстановить все завершенные транзакции из так называемых Masterspace-файлов, содержащих все таблицы, а также из соответствующих файлов логов InnoDB. Это возможно и в случае, если изменения в файле Masterspace не были сохранены. Данная предохранительная мера защищает систему, прежде всего, от потерь информации при аварийном завершении работы или перебоях с электричеством. Кроме того, при этом процесс резервного копирования упрощается. Важно обеспечить лишь «замораживание» файлов

Masterspace и файлов логов InnoDB в определенный момент. Такое «замораживание» гарантируется благодаря мгновенным снимкам LVM.

Создаваемая таким образом резервная копия в двух отношениях отличается от результата работы `mysqldump`. Во-первых, в случае с логическим томом резервное копирование охватывает *все* базы данных, в том числе хранимые процедуры, триггеры, права доступа и т. д. Во-вторых, вся информация сохраняется в двоичной форме. В результате все базы данных могут быть восстановлены только все вместе, а для самого восстановления информации из резервной копии требуется MySQL-сервер такой же конфигурации (и желательно с тем же номером версии), что применялся при создании копии.

Конфигурация. Конфигурация осуществляется в файле `/etc/mylvmbackup.conf`. В следующих строках представлен образец конфигурации. При этом предполагается, что для каталога `/var/lib/mysql` отведен собственный LVM-раздел с именем `/dev/vg1/mysql`.

```
# /etc/mylvmbackup.conf
[mysql]
user         = root
password     = *******
host         = localhost
port         = 3306
socket       =
mycnf        = /etc/mysql/my.cnf
[lvm]
vgname       = vg1
lvname       = mysql
backuplv     =
lvsize       = 5G

[fs]
xfs=0
mountdir     = /var/cache/mylvmbackup/mnt/
backupdir    = /var/cache/mylvmbackup/backup/
relpath      =

[tools]
... (как правило, без изменений)

[misc]
backuptype      = tar
prefix          = backup
tararg          = cvzf
tarsuffixarg    =
rsyncarg        = -avWP
datefmt         = %Y%m%d_%H%M%S
innodb_recover  = 1
pidfile         = /var/tmp/mylvmbackup_recoverserver.pid
```

Несколько замечаний по этому поводу. В `vgname` и `lvname` мы указываем имена VG (группы томов) и LV (логического тома). Из этих данных составляется имя устройства LVM — `/dev/vgname/lvname`. С помощью `backuplv` можно дать разделу

с мгновенным снимком определенное имя. По умолчанию сценарий резервного копирования использует имя mysql_snapshot. Команда lvsize указывает размер буфера для измененных блоков с данными. Буфер должен быть достаточно велик, чтобы в нем уместились все блоки данных из раздела с MySQL, измененные в ходе резервного копирования. Приближенная, но явно завышенная оценка его размера — 5 Гбайт. Наконец, mylvmbackup показывает, сколько памяти фактически используется во время резервного копирования. Располагая такой информацией, можно оптимизировать настройки.

Команда backupdir указывает, где mylvmbackup должна сохранять файлы резервных копий. Желательно, чтобы этот файл находился в ином логическом томе, нежели /var/lib/mysql. Команда relpath определяет, где каталог /var/lib/mysql расположен *относительно* точки, в которой логический том подключается к системе. Если для /var/lib/mysql существует собственный логический том, то команда relpath пуста. Если, напротив, в логическом томе находится *весь* каталог /var, то вы должны использовать relpath=lib/mysql/.

Команда innodb_recover указывает, следует ли уже во время резервного копирования проверить синхронность Masterspace-базы данных InnoDB и файлов логов этой базы данных. Если синхронизация отсутствует, то в рамках резервного копирования происходит восстановление InnoDB, чтобы можно было выполнять транзакции, указанные в файле лога, также и в файле Masterspace. Таким образом, увеличивается время, необходимое для резервного копирования, но сокращается время, которое впоследствии понадобится на настройку системы на базе резервной копии.

Создание резервной копии. Сама резервная копия создается так:

```
root# mylvmbackup
20101128 14:50:23 Info: Connecting to database...
20101128 14:50:23 Info: Flushing tables with read lock...
20101128 14:50:23 Info: Taking position record...
20101128 14:50:23 Info: Taking snapshot...
File descriptor 3 left open
   Logical volume "mysql_snapshot" created
20101128 14:50:24 Info: Unlocking tables...
20101128 14:50:24 Info: Disconnecting from database...
20101128 14:50:24 Info: Mounting snapshot...
20101128 14:50:24 Info: Recovering innodb...
...
```

После этого нужно найти в каталоге /var/cache/mylvmbackup/backup архив *.tar.gz, снабженный указанием актуальной даты. В этом архиве кроме самих файлов базы данных содержится копия my.cnf, а также информация о логировании и репликации (файл backup-pos/*.pos).

Восстановление базы данных. Следующие команды показывают, как возобновить резервное копирование. Первая tar-команда может показаться немного непонятной. Она извлекает лишь те файлы, которые внутри архива расположены в каталоге backup. Затем эти данные записываются в каталог /var/lib/mysql (без указания backup). Вторая tar-команда аналогичным образом извлекает архивные файлы backup-pos/* прямо в каталог /etc/mysql.

```
root# /etc/init.d/mysql stop
root# rm -rf /var/lib/mysql/* (или) mv /var/lib/mysql/* /bak/
```

```
root# mv /etc/mysql/my.cnf /etc/mysql/my.cnf.bak
root# tar -x -f backup.tar.gz -C /var/lib/mysql --strip 1 backup
root# tar -x -f backup.tar.gz -C /etc/mysql --strip 1 backup-pos
root# mv /etc/mysql/backup-*_my.cnf /etc/mysql/my.cnf
root# /etc/init.d/mysql start
```

Инкрементное резервное копирование с применением двоичного логирования

С помощью сценария можно автоматизировать резервное копирование, осуществляемое с помощью `mysqldump`, и делать резервные копии регулярно — ежедневно или еженедельно. Чтобы минимизировать возможную потерю данных в случае аварии, можно также активизировать инкрементное резервное копирование. Для этого вставьте в `/etc/mysql/my.cnf` следующую строку и перезапустите MySQL-сервер.

```
# Изменение в /etc/mysql/my.cnf
log_bin = /var/log/mysql/mysql-bin.log
```

Теперь MySQL-сервер будет протоколировать все команды, которые изменяют данные. Файлы логов автоматически нумеруются по порядку (`mysql-bin.000001`, `.000002` и т. д.). Поскольку всегда изменяется только последний файл, нам относительно несложно с короткими перерывами переносить эти файлы в каталог с резервными копиями (например, с помощью `rsync`).

Если вам потребуется восстановить ваши базы данных, то сначала выполните описанные выше шаги по восстановлению данных для последней полностью проведенной операции резервного копирования. После этого внесите с помощью `mysqlbinlog` все изменения, которые произошли со времени создания этой копии. Последовательность команд выглядит так:

```
root# mysqlbinlog --start-position=<p> mysql-bin.<n> | mysql -u root -p
root# mysqlbinlog mysql-bin.<n+1> | mysql -u root -p
root# mysqlbinlog mysql-bin.<n+2> | mysql -u root -p
..
```

Прежде чем углубиться в работу, нужно узнать, каков будет номер первого файла логов (`<n>`), который мы будем учитывать, и с какой позиции в первом файле (`<p>`) начинать работу. К счастью, `mylvmbackup` сохраняет эту информацию в архиве с резервной копией. Если вы извлекли информацию, как показано выше, то нужные вам данные находятся в двух первых строках файла `/etc/mysql/*_msql.pos`:

```
root# less /etc/mysql/backup-20101128_145023_mysql.pos
Master:File=mysql-bin.000016
Master:Position=98
...
```

Репликация. После того как активизировано двоичное журналирование, остается добавить еще небольшой штрих, чтобы настроить репликацию. Нужно синхронизировать второй MySQL-сервер с первым. Таким образом, у вас в любой момент будет работать вторая активная система с базой данных, в экстренном случае она сможет заменить первую систему. Введение в репликацию баз данных MySQL дается в учебнике по MySQL: http://dev.mysql.com/doc/refman/5.5/en/replication.html.

25 Резервное копирование

Я совсем не пытаюсь убедить вас в необходимости резервного копирования. Основная цель этой главы — рассказать о некоторых инструментах резервного копирования. Какими инструментами пользоваться и в каких сочетаниях — решать вам, общего рецепта не существует. Оптимальная стратегия резервного копирования слишком сильно зависит от природы данных, вида компьютера (ПК/ноутбук/сервер), носителя резервной копии (внешний диск, сетевой каталог, облако) и многих других факторов.

25.1. Программы для резервного копирования с графическим пользовательским интерфейсом

Создав программу Time Machine, компания Apple доказала, что даже программа для резервного копирования может быть захватывающей. Linux в этой области пока достиг гораздо более скромных успехов. Программ и инструментов для резервного копирования существует немало, но пользователи настольных компьютеров так и не дождались пока простого, как все гениальное, инструмента для резервного копирования. Gnome и KDE по умолчанию вообще не содержат такой программы. В этом разделе я кратко расскажу вам о некоторых из подобных программ.

- Déjà Dup — https://launchpad.net/deja-dup;
- Grsync — http://www.opbyte.it/grsync/;
- Back in Time — http://backintime.le-web.org/.

Наиболее популярной из этих программ в настоящее время является Déjà Dup: она по умолчанию устанавливается в Fedora и Ubuntu и применяется в этих дистрибутивах как стандартная программа для резервного копирования. При работе с Grsync довольно сложно ошибиться — программа просто синхронизирует ваши данные в другом каталоге, и резервную копию можно читать без дополнительных инструментов.

СОВЕТ

Подбирая программу для резервного копирования, избегайте изолированных приложений! Если окажется, что разработка либо техническая поддержка вашей программы прекратилась, то вам придется переходить к работе с другим инструментом.

Déjà Dup

Программа Déjà Dup написана таким образом, чтобы вы могли с минимальными сложностями пересохранить домашний каталог в локальном резервном каталоге или в резервном каталоге, доступном по SSH. В Ubuntu программа Déjà Dup находится в системных настройках, и ее можно запускать оттуда. В Fedora нужно открыть обзор программ Gnome и искать программу для резервного сохранения данных (Backup).

Конфигурация. Конфигурация Déjà Dup осуществляется в трех разделах: Storage (Память), Folders (Каталоги) и Schedule (Расписание). В диалоговом окне Save (Сохранить) вы указываете, где должны сохраняться резервные копии. Как правило, в качестве места сохранения резервной копии указывается локальный каталог (Local Folder), в котором можно выбрать любой подкаталог. Если вы хотите использовать внешний диск или флешку не только для резервного копирования, целесообразно создать там отдельный каталог для резервных копий. Целью резервного копирования может быть FTP-каталог, сервер, доступный через SSH, каталог WebDAV, либо сетевой каталог Windows или Samba.

В разделе Folders (Каталоги) вы указываете, какие каталоги должны быть защищены программой резервного копирования. По умолчанию в резервную копию попадает все содержимое вашего домашнего каталога, за исключением корзины и каталога Downloads. Часто бывает целесообразно также исключать из регулярного резервного копирования каталоги с очень крупными файлами (например, с видео или с каталогами виртуальных машин, если вы пользуетесь VirtualBox).

В разделе Schedule (Расписание) вы указываете, как часто должно происходить резервное копирование (ежедневно, еженедельно, ежемесячно и т. д.) Впоследствии Déjà Dup будет запускаться автоматически сразу же после вашего входа в систему и самостоятельно выполнять резервное копирование. Если носитель для резервного копирования в данный момент недоступен, то Déjà Dup откладывает резервное копирование и начинает операцию автоматически, как только носитель окажется доступен. Это очень удобно: вам приходится позаботиться лишь о том, чтобы носитель для резервного копирования регулярно оказывался доступен. Все остальное программа сделает сама.

Кроме того, можно указать, с какой периодичностью программа резервного копирования должна заново сохранять изменяющиеся данные. Наиболее надежна стандартная настройка Always (Всегда), но при ней даже довольно большой носитель для резервных копий будет быстро израсходован. Если задействовать настройку At Least Three Months (Минимум три месяца), то в резервную копию будут попадать *все* данные, но для тех, которые изменяются особенно часто, будут сохраняться лишь версии за последние три месяца.

В диалоговой вкладке Overview (Обзор) обобщаются все настройки. Здесь вы можете либо провести вручную *одну* операцию резервного копирования (Backup Now), либо активизировать параметр Automatic Backups (Автоматические резервные копии). Во втором случае программа сама будет обеспечивать регулярное резервное копирование.

При первом резервном копировании нужно, кроме того, указать, следует ли шифровать резервные копии (и если да — то с каким паролем). Но учитывайте, что

при шифровании потребляется достаточно много дополнительных ресурсов процессора!

Первая операция резервного копирования, осуществляемая Déjà Dup, длится очень долго. При последующих операциях резервного копирования сохраняются лишь изменения, и работа значительно ускоряется.

СОВЕТ

Déjà Dup может выполнять резервное копирование и в системе Amazon S3. В Ubuntu этот вариант по умолчанию на выбор не предоставляется. Чтобы исправить ситуацию, установите пакет python-boto. Довольно своеобразны и стандартные настройки Déjà Dup в Ubuntu. В качестве места для сохранения резервных копий по умолчанию выбирается подкаталог в директории Ubuntu-One. Но эта настройка целесообразна лишь при условии, что вы собираетесь сохранять небольшие объемы данных, а также обладаете очень хорошим соединением с Интернетом (в данном случае решающее значение приобретает скорость закачки).

Восстановление данных. С помощью кнопки Recovery (Восстановление) можно восстановить резервную копию в полном объеме. При этом вы можете выбрать желаемую версию резервной копии и указать, куда должны копироваться файлы резервных копий.

Если вы хотите восстановить более раннюю версию единственного файла, то совсем не обязательно запускать в системных настройках модуль Backup (Резервное копирование). Вместо этого щелкните на файле или каталоге в файловом менеджере Nautilus и выполните команду из контекстного меню Вернуться к более ранней версии. В файловом менеджере можно восстанавливать и удаленные файлы. Команда Восстановить исчезнувшие файлы открывает специальное диалоговое окно, где отображаются данные, для которых сохранены резервные копии, но которые уже отсутствуют в своем исходном каталоге.

Программа Déjà Dup работает на основе специального сценария резервного копирования, который называется Duplicity. Этот сценарий написан на языке Python. Особый недостаток этого сценария заключается в том, что данные сохраняются в очень специфическом формате, так что восстановить их можно только с помощью Déjà Dup или, собственно, Duplicity.

Grsync

На самом деле «инструмент для резервного копирования» — слишком громкое название для Grsync. Это простой графический интерфейс для работы с командой Grsync (рис. 25.1). После установки программы подключите к компьютеру внешний диск или флешку, запустите Grsync и скопируйте ваш домашний каталог в каталог на новом носителе. В первый раз при этом должны быть скопированы все файлы. Впоследствии будут копироваться только измененные или новые файлы.

Различные параметры можно оставить заданными по умолчанию. Но о двух параметрах следует поговорить подробнее.

- Удалить на приемнике — указывает, должна ли Grsync синхронизировать процессы удаления. В таком случае, если после первого резервного копирования вы удаляете в домашнем каталоге какой-либо файл, то при следующем резервном копировании данный файл будет удален и в резервном каталоге. Если ре-

зервную копию требуется защитить от нечаянного удаления, то установить этот флажок нельзя (по умолчанию он также не установлен). Если же вам важно, чтобы содержимое резервной копии точно соответствовало содержимому защищаемого каталога, этот флажок нужно установить.

- Не покидать файловую систему — означает, что Grsync будет синхронизировать лишь такие данные, которые находятся в файловой системе исходного каталога. Как правило, этот флажок должен быть установлен.

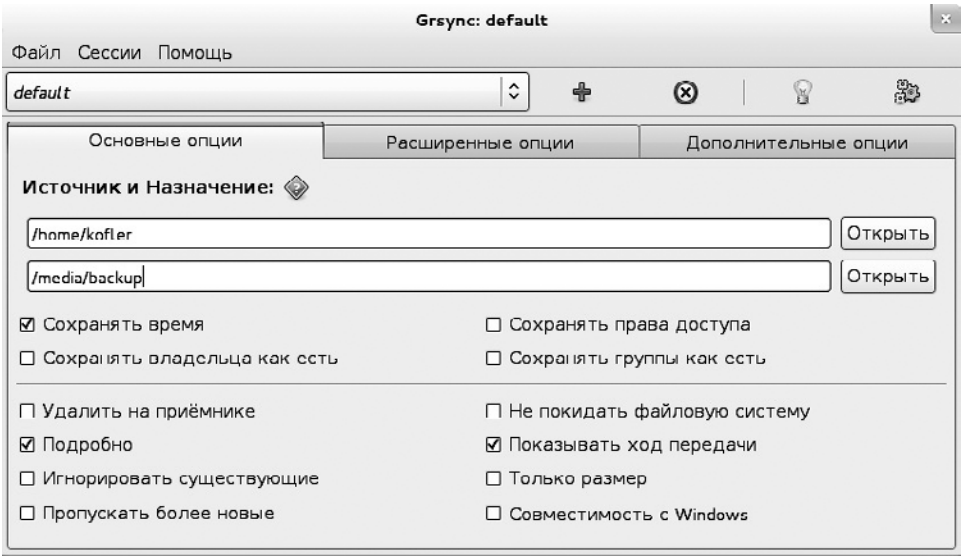


Рис. 25.1. Синхронизация каталогов в Grsync

К основным достоинствам Grsync относятся простота в обращении с программой и то обстоятельство, что ваши файлы переписываются в каталог с резервной копией в соотношении 1:1. Для обращения к вашей резервной копии не требуется никаких специальных инструментов.

Back In Time

Back In Time — это программа для резервного копирования личных данных. И в Gnome, и в KDE для нее предоставляется специальный пользовательский интерфейс. В большинстве современных дистрибутивов Back in Time предоставляется в виде готового пакета. После установки выполните первичную конфигурацию с помощью Alt+F2 `backintime-kde` или `backintime-gnome`.

Настройка осуществляется на шести вкладках.

- General (Общие). Здесь указывается, в каком каталоге должны сохраняться резервные копии (в идеале, этот каталог должен находиться за пределами вашего домашнего каталога). Каталог может располагаться на внешнем носителе данных, который постоянно подключен к вашему компьютеру. Вы должны иметь право изменять файлы в каталоге с резервными копиями.

Кроме того, на этой вкладке задается, как должно производиться резервное копирование. Как правило, нужно указывать Every hour (Ежечасно) или Every day (Ежедневно). Можно также выбрать из списка команду None (Нет). В таком случае резервное копирование придется выполнять вручную. Это бывает целесообразно, например, когда вы применяете для резервного копирования внешний диск, но данный диск не всегда подключен к компьютеру.

- Include (Включить). Здесь вы выбираете, для каких каталогов должны создаваться резервные копии. Обычно здесь следует просто указать ваш домашний каталог. Но можно сделать и другой выбор, например сохранять резервные копии каталогов Мои документы или Мои рисунки.
- Exclude (Исключить). Здесь вы указываете, какие (под)каталоги и типы файлов исключаются из резервного копирования (например, Загрузки). По умолчанию, в программе Back in Tine предусмотрено, что резервные копии *не* создаются для скрытых файлов и каталогов. Это довольно опасная настройка, так как именно в скрытых файлах зачастую находится важная информация о приложении (например, в каталоге ~/.thunderbird располагаются ваши электронные письма, если вы используете Thunderbird в качестве почтового клиента).
- Auto-remove (Автоматически удалять). Чтобы объем резервных копий не рос совершенно бесконтрольно, здесь можно указать, какая информация из резервных копий должна автоматически удаляться. Обычно подходит параметр Smart remove (Интеллектуальное удаление). При установке данного флажка вчерашние резервные копии никогда не удаляются. Более старые копии, в основном, удаляются. При этом программа гарантирует, что сохраняется по одной резервной копии за последние две недели, за каждый месяц текущего года и за каждый прошедший год.
- Options (Настройки) и Expert Options (Экспертные настройки). Здесь находятся настройки для опытных пользователей. Изменять их, как правило, не требуется.

После завершения конфигурации появляется графический интерфейс программы Back in Time. Здесь вы в любой момент можете вручную запустить резервное копирование (в том числе, отклоняясь от заданной периодичности), а также давать имена имеющимся резервным копиям.

Восстановление данных. Важнейшая функция программы с пользовательским интерфейсом заключается в том, чтобы вы могли, пользуясь простым обозревателем каталогов, в любой момент получить доступ к любым данным, для которых сохранены резервные копии. С помощью кнопки Recover (Восстановить) вы можете восстановить файлы, которые были по ошибке удалены или изменены. Команда Copy (Копировать) копирует выбранные файлы. Теперь с помощью файлового менеджера вы можете вставить эти файлы куда угодно.

По умолчанию программа Back in Time может сохранять лишь личные данные. Если вы хотите использовать ее для резервного копирования системной информации, то программу нужно запустить в административном режиме (root). Как в Gnome, так и в KDE предусмотрен такой вариант запуска.

Внутренняя организация. «За кулисами» за автоматическое выполнение резервного копирования отвечает Cron. Эта задача выполняется командой `backintime`.

При этом интерпретируется конфигурационный файл `~/.config/backintime/config`. Пользовательский интерфейс Back in Time не должен работать во время резервного копирования! Управление Cron осуществляется в файле `/var/spool/cron/tabs/`*`логин`*.

В резервном каталоге все файлы копий сохраняются в несжатом виде. (К сожалению, параметр сжатия в программе отсутствует). Файлы, которые не изменяются от одного резервного копирования до следующего, связываются друг с другом лишь с помощью так называемых жестких ссылок (hard link). В результате на диске экономится очень много места. В любом случае подобный механизм действует лишь на таких носителях данных, которые его поддерживают (то есть не подходят жесткие диски и флешки с FAT-форматированием).

25.2. Резервное копирование на NAS-устройствах

Если вы работаете только с одним компьютером, то идеальным носителем для резервных копий будет внешний жесткий диск или флешка. Но если применяется одновременно несколько компьютеров, то зачастую требуется центральное место для резервного копирования и возможность обмена файлами между машинами. Для этого лучше всего подходят NAS-устройства. Аббревиатура NAS расшифровывается как Network Attached Device (сетевое устройство для хранения данных). Это носитель, доступ к которому открыт по сети.

Практически на всех NAS-устройствах, имеющихся на рынке, работает система Linux, файловый сервер Samba и веб-сервер, который помогает конфигурировать устройство через более или менее удобный веб-интерфейс. На многих моделях также предоставляется NFS-сервер, FTP-сервер, AFP-сервер (для Apple), SSH-сервер, клиенты для потоковой передачи мультимедиа (Multimedia Streaming), загрузочные инструменты и т. д.

Резервное копирование в сетевых каталогах Windows

Хотя по своей внутренней организации NAS-системы являются Linux-системами, благодаря Samba они работают точно как системы Windows, предоставляющие каталоги в совместное использование. И именно здесь возникает проблема. Как правило, устройства с Linux, предназначенные для резервного копирования, требуют, чтобы резервное копирование осуществлялось в файловой системе, совместимой с UNIX или Linux. Но в каталоге Windows невозможно сохранить права доступа, отвечающие правилам UNIX. Кроме того, каталоги резервного копирования Linux, как правило, не имеют прямого доступа к сетевым каталогам Windows.

Существует несколько возможностей решить эти проблемы.

- Чтобы инструменты резервного копирования могли записывать информацию в сетевой каталог Windows, этот каталог необходимо предварительно подключить к файловой системе компьютера Linux. Это можно сделать либо

в диалоговом окне файлового менеджера, либо путем изменения системного файла /etc/fstab.

- Чтобы избежать потери информации о правах доступа к Linux в ходе резервного копирования, можно упаковать копируемые файлы, например в архив TAR. Описанная выше программа Déjà Dup действует именно так.
- Кроме того, можно попытаться сохранять копируемые данные прямо в сетевом каталоге Windows. Это делается командой rsync или соответствующей программой с графическим пользовательским интерфейсом Grsync. Если вам повезет, то сохранятся даже биты доступа ваших файлов (в частности, бит Execute в исполняемых файлах программ или сценариев). Это происходит и в тех случаях, когда сетевой каталог предоставляется в использование не компьютером с Windows, а достаточно новым сервером Samba, который поддерживает расширения POSIX и передает их через CIFS (общую файловую систему Интернета). Но, по моему опыту, этот механизм зачастую не функционирует, так как на многих устройствах с NAS работают слишком старые версии Samba.
- Наконец, на некоторых NAS-устройствах предоставляется возможность использовать в качестве носителя резервных копий и NFS-каталоги или Rsync-сервер. Но в зависимости от устройства конкретная конфигурация бывает довольно сложной.

Mount или /etc/fstab

Насколько я могу судить, самый надежный способ резервного копирования на NAS-устройстве связан с использованием команды mount или записи в файле /etc/fstab. Цель заключается в том, чтобы подключить каталог NAS-устройства к дереву каталогов локального компьютера. Этот метод подробно описан в разделе 20.7. Однако рассмотренный метод предполагает, что NAS-устройство все время должно работать и сетевой каталог всегда должен быть доступен.

Подключение сетевого каталога Windows к файловой системе с помощью Gnome

Чтобы не выполнять команду mount вручную и не изменять /etc/fstab, сетевой каталог можно просто подключить к локальной файловой системе через Gnome. Для этого выполните File ▸ Connect with Server (Файл ▸ Подключить к серверу), выберите тип службы Windows Share (Совместная работа в Windows), а потом укажите хост-имя сервера, имя предоставляемой раздачи, при необходимости — имя находящегося там каталога, а также логин и доменное имя.

Кроме того, установите флажок Enable bookmarks (Активизировать закладки), чтобы впоследствии эти данные не пришлось вводить перед началом каждого сеанса резервного копирования. Закладка впоследствии будет отображаться в меню Locations (Места), а также в диалоговых окнах для выбора файлов в Gnome. Как только вы нажмете Connect (Соединить), система запросит у вас пароль к сетевому каталогу Windows. Gnome может сохранять этот пароль — так многократное резервное копирование становится гораздо удобнее.

«За кулисами» для доступа к сетевым каталогам Gnome использует GVFS (Виртуальную файловую систему Gnome). Интеграция сетевых каталогов в дерево локальной системы осуществляется в невидимом каталоге .gvfs. В Nautilus не составляет труда найти и использовать подключенные таким образом каталоги, но, к сожалению, это справедливо не для всех программ резервного копирования. Так, Déjà Dup отображает сетевые каталоги непосредственно в диалоговом окне для выбора файлов, а в Grsync сетевой каталог нужно явно указывать именно как подкаталог .gvfs.

25.3. Сжатие и архивация файлов

В следующих разделах я представлю целую палитру команд, которые помогают архивировать и защищать файлы. Это `tar`, `zip`, `rsync`, `rdiff-backup`, `rsnapshot` и т. д. Начнем разговор с команд для сжатия и архивации. Они приведены в табл. 25.1.

Таблица 25.1. Инструменты для сжатия и архивации файлов

Команда	Значение
gzip	Сжимает файл
gunzip	Восстанавливает файл в прежнем размере
bzip2	Сжимает файл (сжатие сильнее, чем в случае с gzip, но работает медленнее)
bunzip2	Восстанавливает файл в прежнем размере
xz	Сжимает файл (сжатие еще более сильное, чем в случае с bzip2, но работает еще медленнее)
unxz	Восстанавливает файл в прежнем размере
lzop	Сжимает/восстанавливает файлы значительно быстрее, чем gzip
tar	Создает либо распаковывает файловый архив
zip	Создает Windows-совместимый ZIP-архив
unzip	Извлекает zip-архив
zipinfo	Отображает информацию по ZIP-архиву

Сжатие файлов (gzip, bzip2, xz, lzop)

Gzip и gunzip. `Gzip` сжимает файлы, указанные в виде параметров и переименовывает их по принципу `name.gz`. Команда `gunzip` функционирует в обратном направлении. Обе команды используют LZ77 — алгоритм Лемпеля — Зива, который особенно хорош для работы с текстовыми файлами (но не для аудио- и видеофайлов). Сжатие, разумеется, происходит без потерь, то есть после восстановления данных из архива исходные данные получаются такими же, какими были до сжатия. Далее показано применение этих механизмов:

```
user$ ls -l filesystem.tex
... 178794 1. Aug 17:43 filesystem.tex
user$ gzip filesystem.tex
user$ ls -l filesystem.tex.gz
... 57937 1. Aug 17:43 filesystem.tex.gz
user$ gunzip filesystem.tex.gz
```

Bzip2 и bunzip2. Команды bzip2 и bunzip2 — это альтернативы для gzip/gunzip. Их преимущество заключается в более качественном сжатии, недостаток — в более медленном исполнении. Файлы, сжатые таким образом, имеют расширение .bz2.

```
user$ bzip2 filesystem.tex
user$ ls -l filesystem.tex.bz2
... 47105 1. Aug 17:43 filesystem.tex.bz2
user$ bunzip2 filesystem.tex.bz2
```

Xz и unxz. Вместо gzip и bzip2 вы можете также использовать для сжатия xz. В итоге файлы получаются еще более маленькими, чем в двух предыдущих примерах. Соответственно, работа программы требует еще больше времени и ресурсов процессора. Если ваша основная цель — сжать файл до минимально возможного размера, можете попробовать команду 7zr из пакета p7zip.

Lzop. Другая цель ставится при использовании команды lzop. Она работает гораздо быстрее, чем все вышеупомянутые. Но результирующие файлы получаются сравнительно крупными (мои тесты показали размер примерно на 50 % больше, чем после обработки gzip). Применение lzop рекомендуется, прежде всего, в тех случаях, когда сжатие должно осуществляться на лету при минимально возможной нагрузке на процессор. Например, такая ситуация возникает при передаче больших объемов данных по сети.

В следующем примере логический том считывается с помощью cat и сжимается с использованием lzop. На это уходит лишь немногим больше времени, чем при прямом копировании логического тома в файл образа.

```
root# cat /dev/vg1/lv3 | lzop -c > lv3.img.lzo (55 секунд)
root# cat /dev/vg1/lv3 > lv3.img               (50 секунд)
```

Создание сжатых архивов (tar, zip)

Tar. Команда tar лучше всего в Linux подходит для сбора нескольких файлов в архиве. Сжатие архива при этом обычно происходит с помощью gzip или bzip2. Изначально tar предназначалось для того, чтобы записывать данные на стример (ленточный накопитель) или считывать их со стримера. Поскольку в наше время стримеры применяются лишь изредка, мы поговорим здесь только о работе с файловым архивом.

Следующая команда добавляет все файлы из каталога buch в сжатый файл buch.tgz.

```
user$ tar -czf mojarchiv.tgz buch/
```

Кратко расскажу о буквенных обозначениях параметров.

- c означает create (создать), то есть команда tar должна создать архив.
- z означает zip (сжать). Таким образом, архив должен быть сжат командой gzip.
- f означает file (файл). То есть tar должна заархивировать файл (а не записать его на кассету стримера). Желаемое имя файла указывается после параметра. Такие архивы обычно имеют расширение .tar.gz или просто .tgz.

Команда `tar -tzf` выдает содержание архива. Файлы внутри архива расположены произвольно. В большинстве дистрибутивов команда `less` сконфигурирована так, что вы можете просмотреть содержимое большинства архивов с помощью `less name.tgz`.

```
user$ tar -tzf mojarchiv.tgz
linuxbuch/
linuxbuch/lanserver.tex
linuxbuch/security.tex~
linuxbuch/buch.tex
linuxbuch/u4.txt~
...
```

Команда `tar -xzf` распаковывает архив и извлекает все содержащиеся в нем файлы.

```
user$ cd drugaja-papka/
user$ tar -xzf mojarchiv.tgz
```

В следующем примере `tar` извлекает из архива только TEX-файлы. Обратите внимание на апостроф в имени — шаблоне файла. Он нужен, чтобы избежать немедленной интерпретации файла оболочкой!

```
user$ tar -xzf mojarchiv.tgz '*.tex'
```

Если вы собираетесь сжимать архивы с использованием `bzip2`, а не с помощью `gzip`, замените параметр `z` на `j`.

Zip. В мире Linux/UNIX TAR является предпочтительным форматом передачи файлов в архиве. Но если вы обмениваетесь информацией с пользователями Windows, то лучше воспользоваться ZIP-архивами.

Следующая команда добавляет в архив `mojarchiv.zip` все HTML-файлы, указанные в качестве параметров:

```
user$ zip mojarchiv.zip *.html
```

Если вы собираетесь архивировать содержимое целых каталогов, укажите параметр `-r`:

```
user$ zip -r mojarchiv.zip mojwebsite/
```

Чтобы просмотреть содержимое ZIP-файла, используйте `zipinfo`:

```
user$ zipinfo mojarchiv.zip
Archive: test.zip 143677915 bytes 1899 files
-rw-r--r-- 2.3 unx  78039 tx defN 10-Jul-06 11:27 linuxbuch/lanserver.tex
-rw-r--r-- 2.3 unx 115618 tx defN 7-Apr-05 15:58  linuxbuch/security.tex~
-rw-r--r-- 2.3 unx   3899 tx defN 28-Jul-06 16:38 linuxbuch/buch.tex
-rw-r--r-- 2.3 unx    752 tx defN 11-Feb-04 12:06 linuxbuch/u4.txt~
...
```

Для распаковки архива применяется `unzip`:

```
user$ cd drugaja-papka/
user$ unzip mojarchiv.zip
```

25.4. Синхронизация каталогов (rsync)

Команда rsync синхронизирует деревья каталогов. Таким образом, при ее первом прогоне можно скопировать все файлы из одного каталога в другой. При последующих прогонах будут копироваться лишь измененные файлы и (по желанию) будет выполняться репликация процессов удаления. Команда rsync отлично подходит для того, чтобы (например, ежедневно или еженедельно) синхронизировать полную копию определенного каталога на внешнем диске.

rsync может использоваться и через сетевое соединение. По умолчанию обмен информацией происходит через SSH. При этом передаваемые данные одновременно шифруются. В качестве альтернативы можно применять другую внешнюю shell-программу либо сконфигурировать на противоположной стороне второй rsync-сервер.

В табл. 25.2 приведен синтаксис, используемый при указании исходного и целевого каталогов. В табл. 25.3 обобщены важнейшие параметры для управления rsync.

Таблица 25.2. Указание исходного и целевого каталогов rsync

Запись	Значение
file1 file2	Локальные файлы
directory	Локальный каталог
host:dir	Каталог на компьютере host
user@host:dir	Как было показано выше, вход под логином SSH и именем user
rsync://user@host/dir	Обмен информацией с rsync-сервером
rsync://user@host:port/dir	Rsync-сервер на заданном порте

Таблица 25.3. Параметры rsync

Параметр	Действие
-a или --archive	Рекурсивно копирует и сохраняет всю информацию о файлах
--delete	Удаляет в целевом каталоге файлы или каталоги, которые уже не существуют в исходном каталоге
-D	Учитывает все файлы устройств и специальные файлы
--exclude=pattern	Пропускает указанные файлы
-g или --group	Получает групповую принадлежность
-l или --links	Дублирует символьные ссылки
-o или --owner	Получает информацию о владельце файла
-p или --perms	Получает права доступа
-r или --recursive	Рекурсивно копирует все подкаталоги
-t или --times	Получает время осуществления изменений
-u или --update	Игнорирует обрабатываемые сейчас, старые файлы
-v или --verbose	Показывает, что в данный момент происходит
-W или --whole-file	При изменениях копирует весь файл
-z	Сжимает информацию, переданную по SSH

Локальное применение. Чтобы синхронизировать целый каталог, включая все его подкаталоги, используйте параметр -a, который является сокращенным вари-

антом записи нескольких других параметров (`-rlptgoD`). Данный параметр обеспечивает рекурсивную обработку всех подкаталогов и гарантирует, что сохранится максимально возможное количество информации о файлах (сведения о владельце, групповая принадлежность, время внесения последних изменений и т. д.). Если каталог dir2 пока не существует, то он создается. В отличие от случая с `cp`, не копируются файлы, которые остались без изменений с момента последнего копирования.

```
user$ rsync -a dir1/ dir2/
```

По умолчанию `rsync` копирует и, соответственно, обновляет все новые или измененные файлы, но ничего не удаляет. Если вы хотите, чтобы файлы, удаляемые из каталога dir1, также удалялись и из каталога dir2, дополнительно укажите параметр `--delete`. Очевидно, что этот параметр довольно опасен: если вы случайно удалите каталог, то именно данный каталог будет удален на резервном диске при следующем запуске программы!

Применение в сети. При применении `rsync` для синхронизации каталогов на различных компьютерах следует указывать исходный и целевой каталоги по принципу `hostname: directory`, либо `username@hostname:directory`. В первом случае `rsync` использует актуальное имя пользователя.

Благодаря следующей команде, каталог dir1 локального пользователя `username` с компьютера `saturn.sol` синхронизируется с каталогом dir2 с компьютера `mars.sol`. За ввод пароля отвечает `ssh`. (Итак, нужно указать пароль пользователя `username` с компьютера `mars.sol`.)

```
username@saturn.sol$ rsync -e ssh -az dir1/ mars.sol: dir2/
username@mars.sol's password: ******
```

Команда `rsync` может переносить файлы с удаленного компьютера на локальный. Следующая команда выполняет синхронизацию в обратном направлении:

```
username@saturn.sol$ rsync -e ssh -az mars.sol:dir2/ dir3/
username@mars.sol's password: ******
```

Если `rsync` должна вызываться автоматическим сценарием резервного копирования, то интерактивный ввод пароля, конечно, неуместен. Решение заключается в том, чтобы создать на локальном компьютере файл с закрытым ключом, а на парном компьютере — соответствующий файл с открытым ключом. Если при создании ключа вы откажетесь от использования фразы-пароля, то вход под SSH станет возможен без пароля. Но из соображений безопасности нужно создать для сценария резервного копирования собственный аккаунт.

Rsync-сервер. `Rsync` можно использовать не только в комбинации с SSH, как было показано выше, но и настраивать обмен данными между локальной командой `rsync` и `rsync`-сервером на удаленном компьютере. При этом предполагается, что на парном компьютере настроен `rsync`-сервер. Его конфигурация осуществляется в файле `/etc/rsyncd.conf`. В таком случае обмен информацией между `rsync`-клиентом и `rsync`-сервером происходит через порт 873. Поэтому данный порт не должен блокироваться брандмауэром.

На практике конфигурировать `rsync`-сервер рекомендуется лишь в случаях, когда сервер регулярно отражается различными клиентами (то есть работает

в качестве зеркального сервера) (http://unix.stackexchange.com/questions/26182/ и http://serverfault.com/questions/100707/rsync-daemon-is-it-really-useful).

25.5. Инкрементные резервные копии (rdiff-backup)

Интересной альтернативой для rsync является команда rdiff-backup. Ее важнейшее отличие от rsync заключается в том, что при работе с измененными файлами rdiff-backup архивирует в резервном каталоге и старую версию файла. Для экономии места можно сохранять не копию конкретного файла, а только внесенные изменения (можно в сжатой форме). Кроме того, rdiff-backup позволяет без особого труда организовать инкрементное копирование. Из такой копии, сохраняемой с постепенным приращением, можно восстановить и более старые версии файла. В принципе rdiff-backup предлагает те же функции, что и Time Machine из Apple OS X — просто без шикарного пользовательского интерфейса.

Резервное копирование. Проще всего применять rdiff-backup к двум локальным каталогам. Если целевой каталог пока не существует, то программа его создает.

```
root# rdiff-backup /home /home-backup
```

В каталоге резервного копирования rdiff-backup создает подкаталог rdiff-backup-data. Здесь программа сохраняет различные статистические данные и статусную информацию. Кроме того, в каталоге increments содержатся старые версии файлов, которые после сохранения данной версии были изменены или удалены. При этом сохраняются только изменения (.diff), которые дополнительно сжимаются. В имя файла записывается и дата создания последней версии. Получаются довольно длинные имена файлов, в форме filename.2010-04-03T08:37:58+02:00.diff.gz.

Доступ к резервным копиям. Если вы хотите получить доступ к резервной копии, то обратитесь к каталогу /home-backup. Здесь содержится каталог /home в том состоянии, в котором он находился на момент последнего резервного копирования. (Если не считать дополнительных подкаталогов rdiff-backup-data — он пребывает именно в таком состоянии, как если бы резервная копия создавалась с помощью cp -a или rsync -a --delete.) Доступ к последней резервной копии также не составляет труда. Разумеется, эту копию можно просто восстановить с помощью rdiff-backup. Для этого применяется параметр -r и указание времени now. Следующая команда, например, восстанавливает резервную копию во временном каталоге.

```
root# rdiff-backup -r now /home-backhup /tmp/home-actual
```

Если вы хотите обратиться к более ранней версии файла или к удаленному файлу, то дело обстоит сложнее. Нужно по порядку использовать все .diff-файлы (начиная с самого нового), пока не дойдете таким образом до желаемой более ранней версии. Конечно, вам не придется заниматься этим вручную — воспользуйтесь командой rdiff-backup. Следующая команда восстанавливает состояние каталога /home в таком виде, в каком он находился десять дней назад:

```
root# rdiff-backup -r 10D /home-backup/ /tmp/home-historic
```

Время резервного копирования может быть задано как в абсолютном виде (например, 2010-12-31), так и в относительном виде — в часах (h), днях (D), неделях (W) и т. д. См. также раздел Time Formats (Форматы времени) в справочном файле man rdiff-backup. Учитывайте, что при восстановлении старых файлов с возрастающим номером версии процессор испытывает значительную нагрузку и, соответственно, работает медленно.

Часто требуется восстановить старую версию только одного файла или каталога. При этом можно указать даже такой файл или каталог, который уже удален:

```
root# rdiff-backup -r 10D /home-backup/file file-historic
root# rdiff-backup -r 10D /home-backup/dir/ dir-historic
```

Удаление всех резервных копий. Если вы регулярно выполняете rdiff-backup, то объем каталога с резервными копиями со временем только увеличивается. Чтобы удалить все файлы резервных копий, которые старше четырех месяцев, действуйте так:

```
root# rdiff-backup --remove-older-than 4M --force /home-backup/
```

Можно задать не конкретный момент времени, а указать, какое максимальное количество резервных версий должно оставаться в архиве. Приведенная ниже команда уменьшает количество резервных версий до трех (независимо от их возраста):

```
root# rdiff-backup --remove-older-than 3B --force /home-backup/
```

Сетевое резервное копирование. Во всех предыдущих примерах я исходил из того, что и исходный, и целевой каталоги находятся в локальной файловой системе. Но rdiff-backup может работать и по сети, обращаясь к внешним каталогам. Однако для этого rdiff-backup, в отличие от rsync, должна быть установлена и на внешнем компьютере. Обмен информацией происходит по SSH. Специальной конфигурации rdiff-backup не требуется.

При указании внешних каталогов применяется почти такой же синтаксис, как и при работе с rsync. Единственное отличие заключается в том, что после хост-имени нужно ставить *два* двоеточия:

```
root# rdiff-backup user@firma-abc.de::/home /home-backup
```

Более подробное описание работы с rdiff-backup и примеры приводятся на сайте http://www.nongnu.org/rdiff-backup/.

Duplicity. Если концепция rdiff-backup вас в принципе устраивает, но вы также хотите задействовать шифрование резервной копии и обеспечить закачку по SSH или FTP на внешний сервер, то обратите внимание на программу Duplicity, написанную на языке Python (http://duplicity.nongnu.org/). Работа с ней напоминает использование rdiff-backup, она создает TAR-архивы. Правда, уже много лет Duplicity существует в бета-версии. Тем не менее, она лежит в основе программы Déjà Dup, предназначенной для резервного копирования и снабженной графическим пользовательским интерфейсом. О ней мы говорили в начале этой главы.

25.6. Инкрементные резервные копии (rsnapshot)

На основе rsync также построен сценарий rsnapshot, написанный на языке Perl и находящийся в одноименном пакете. В отличие от описанной выше команды rdiff-backup, эта программа использует жесткие ссылки, позволяющие обращаться к уже защищенным файлам из более ранних резервных копий. Таким образом, доступ к более ранним версиям резервных копий (мгновенным снимкам) выполняется проще, чем с rdiff-backup. Но сжатие такой резервной копии не предусмотрено.

Команда rsnapshot построена так, что сценарий регулярно выполняется автоматически. При соответствующей конфигурации rsnapshot защищает как локальные каталоги, так и каталоги с других компьютеров локальной сети, доступные по SSH. Все резервные копии сохраняются на локальном компьютере — на том, где выполняется rsnapshot. Этот подход прямо противоположен принципу работы многих других инструментов резервного копирования: rsnapshot совершенно не предназначен для сохранения резервных копий локальных файлов на других компьютерах. Команда действует противоположным образом и сохраняет в локальной системе данные с многочисленных компьютеров, доступных по SSH или rsync.

Конфигурация

Команда rsnapshot не получает параметров, а управляется в конфигурационном файле /etc/rsnapshot.conf. В этом файле действуют два важных синтаксических правила. Во-первых, название каталога должно заканчиваться слэшем (например, /directory/, а не /directory). Во-вторых, элементы конфигурационного файла должны отделяться друг от друга знаками табуляции (а не пробелами)!

При первых экспериментах вы можете оставить конфигурационный файл, поставляемый с rsnapshot, почти без изменений. Три важнейших параметра, которые вам известны и которые вы при необходимости можете настроить, — это snapshot_root, backup и interval.

Snapshot_root указывает, в каком каталоге должны сохраняться резервные копии. По умолчанию применяется каталог /var/cache/rsnapshot, автоматически создаваемый при установке rsnapshot.

Backup указывает, где должна сохраняться резервная копия конкретного каталога. backup можно указывать несколько раз (каждый раз — в новой строке), чтобы сохранять несколько каталогов в различных местах. Для резервного копирования в локальной файловой системе настройки backup выглядят следующим образом:

```
# в /etc/rsnapshot.conf
...
backup    /home/ localhost/
backup    /etc/  localhost/
```

Чтобы сделать резервную копию каталога /home/user/Mail с внешнего компьютера mars.sol, доступного по SSH, нужно убрать знак комментария перед уже имеющейся строкой cmd_ssh. В строке backup указывается имя логина, хост-имя,

а также полное название каталога, который требуется защитить. Чтобы этот механизм работал, перед созданием первой резервной копии нужно скопировать в файл /home/user/.ssh/authorized_keys компьютера mars.sol SSH-ключ, созданный с правами администратора. Это делается без пароля. Внимание: файл ключа, сохраненный на локальном компьютере в файле /root/.ssh, не должен попасть в чужие руки!

```
# в /etc/rsnapshot.conf
...
cmd_ssh   /usr/bin/ssh
...
backup    user@mars.sol:/home/user/Mail/ mars.sol/
```

В ходе резервного копирования вы можете создать мгновенный снимок с помощью LVM, выполнять сценарии, исключать из резервного копирования файлы, соответствующие определенному шаблону и т. д. Подробнее об этом можно прочитать в поставляемом вместе с программой конфигурационном файле, а также на странице справки man rsnapshot.

Interval задает, сколько резервных копий за определенный временной период должно сохраняться. Стандартные настройки выглядят так:

```
# в /etc/rsnapshot.conf
...
interval     hourly   6
interval     daily    7
interval     weekly   4
#interval    monthly  3
```

Это означает, что если резервные копии создавались командой rsnapshot hourly, то должно сохраняться по шесть версий таких копий. Далее указано, что будут архивироваться семь резервных копий, полученных в результате rsnapshot daily, а также четыре резервные копии rsnapshot weekly. Интервал monthly по умолчанию не определяется.

Настройка interval hourly 6 целесообразна в том случае, если команда rsnapshot hourly выполняется не ежечасно, а раз в четыре часа (как предусмотрено в /etc/cron.d/rsnapshot). Если же вы действительно собираетесь выполнять команду rsnapshot hourly ежечасно, то лучше задействовать настройку interval hourly 24.

При необходимости можно задать сколько угодно других интервалов. Команда rsnapshot сохраняет резервные копии по каждому интервалу в специально созданном каталоге (для этого интервала). Ссылками соединяются неизмененные резервные копии лишь в рамках одного интервала. Это означает, что при указании каждого дополнительного интервала потребление памяти сильно возрастает.

Вызов вручную

Команду rsnapshot можно вызвать вручную (требует прав администратора). Все параметры нужно указать в файле настройки с определенным временным интервалом:

```
root# rsnapshot daily
```

После резервного копирования вы найдете сохраненные данные в следующем каталоге:

```
/var/cache/rsnapshot/<inverval.n>/<hostname>/<directory>
```

При этом `interval` указывает интервал (по умолчанию — `hourly`, `daily`, `weekly` или `monthly`). Параметр `n` задает номер версии резервной копии (`0` — наиболее актуальная копия, `1` — копия, сохраненная последней, `2` — копия, сохраненная предпоследней и т. д.). Параметр `hostname` указывает, с какого компьютера пришли защищаемые данные (`localhost` соответствует локальному компьютеру).

Наиболее актуальная копия локального каталога `/etc` (не старше часа) находится в следующем каталоге:

```
/var/cache/rsnapshot/hourly.0/localhost/etc
```

Наиболее актуальная копия каталога `/home/user/Mail` с сервера `mars.sol` (не старше месяца) находится в следующем каталоге:

```
/var/cache/rsnapshot/monthly.0/mars.sol/home/user/Mail
```

Автоматический вызов

Чтобы автоматизировать резервное копирование, нужно регулярно вызывать `rsnapshot` с помощью конфигурационного файла Cron `/etc/cron.d/rsnapshot`. Для этого необходимо просто убрать знаки комментариев перед уже имеющимися строками Cron:

```
# /etc/cron.d/rsnapshot
0 */4    * * * root     /usr/bin/rsnapshot hourly
30 3     * * * root     /usr/bin/rsnapshot daily
0 3      * * 1 root     /usr/bin/rsnapshot weekly
30 2     1 * * root     /usr/bin/rsnapshot monthly
```

Таким образом, `rsnapshot` будет срабатывать ежедневно каждые четыре часа (0:00, 4:00, 8:00 и т. д.), ежедневно в 3:30, по понедельникам каждую неделю в 3:00, а также ежемесячно в первый день месяца в 2:30. Программа будет выполняться с параметрами `hourly`, `daily`, `weekly` и `monthly`. Конечно, вы можете варьировать эти сроки по собственному желанию. Но учитывайте, что все временные интервалы, применяемые в Cron-файле, также должны быть определены и в `/etc/rsnapshot.conf`! Интервал `monthly` по умолчанию обычно не задается.

25.7. Сценарии резервного копирования

В основе последнего раздела этой главы лежат практические примеры.

Автоматизация rsync с помощью Cron

Следующий сценарий синхронизирует содержимое локального каталога `data` с одноименным каталогом на внешнем жестком диске под именем `backup`. Сценарий

проверяет, доступен ли внешний диск. Если он недоступен, то резервное копирование не производится.

```
#!/bin/bash
if [ -d /media/backup/ ]; then
   rsync -avW --delete /home/kofler/data /media/backup/
fi
```

Как правило, сценарии резервного копирования выполняются регулярно. Для этого лучше всего создать специальную задачу Cron. Проще всего сохранить сценарий прямо в каталоге Cron, например под именем `/etc/cron.daily/mybackup`. Учитывайте, что в имени файла не должно содержаться точек и что файл должен быть исполняемым (`chmod a+x`).

Другой вариант заключается в том, чтобы сохранить сценарий резервного копирования в произвольном каталоге, например в `/etc/myscripts`. Для автоматического вызова создайте новый файл в `/etc/cron.d`, например, по следующему образцу:

```
# Файл /etc/cron.d/mybackup
15 2 * * * root /etc/myscripts/mybackup
```

Таким образом сценарий резервного копирования будет ежедневно выполняться в 2:15.

Ежедневное и ежемесячное резервное копирование

Следующий сценарий создает сжатый TAR-архив каталога `data`. Архив сохраняется под именами `mydata-day-`*`dd`*`.tar.gz` (ежедневно) и `mydata-month-`*`mm`*`.tar.gz` (ежемесячно). Здесь в *dd* указывается день (с 01 до 31), в *mm* — месяц (с 01 до 12). Если сценарий выполняется ежедневно, то у нас постепенно накопятся 43 резервные версии, отражающие состояние каталога с резервными копиями за последние дни (от 28 до 31 за месяц), а также за последние 12 месяцев.

```
#!/bin/bash
fname1=/backup/mydata-day-$(date "+%d").tar.gz
fname2=/backup/mydata-month-$(date "+%m").tar.gz
tar czf $fname1 /home/kofler/data
cp $fname1 $fname2
chmod 600 $fname1 $fname2
```

Резервное копирование с помощью LVM

Если в ходе резервного копирования файлы изменяются, то результирующая резервная копия получается дефектной. Но на время резервного копирования далеко не всегда бывает возможно остановить работу всех программ и серверных служб. Возможное решение этой проблемы заключается в том, чтобы создать в начале резервного копирования мгновенный снимок LVM и использовать его в качестве основы для резервного копирования. Разумеется, при этом необходимо, чтобы

каталог с защищаемыми данными находился в такой файловой системе, которая сохраняется в логическом томе (а не непосредственно на жестком диске).

Для создания мгновенного снимка применяется команда `lvcreate` с параметром `-s`. С помощью `-L` вы указываете максимальное количество файлов, которые могут быть изменены за период существования мгновенного снимка, то есть за время изготовления резервной копии. В пуле памяти LVM (то есть в физическом томе) для этого должно быть достаточно свободного места. Необходимый размер буферной памяти сначала оценить сложно. Когда резервная копия будет готова, вы сможете узнать с помощью `lvdisplay`, на сколько процентов буфер израсходован в настоящий момент.

В приведенном ниже примере сценария я исхожу из того, что каталог `/home` находится в логическом томе `/dev/vg1/myhome`. Чтобы сделать правильную резервную копию из каталога `/home` и чтобы в ходе работы не изменились никакие файлы, создайте с помощью `lvcreate` мгновенный снимок `homesnap`. Объем буферной памяти составляет 2 Гбайт.

Этот мгновенный снимок подключается к дереву каталогов в каталоге `/var/homesnap`. В дальнейшем мы будем использовать этот снимок в качестве источника данных для команды или сценария резервного копирования. Наконец, `homesnap` извлекается из файловой системы. Снимок удаляется из файловой системы с помощью `lvremove`. Параметр `-f` отменяет запрос о подтверждении этого действия.

```
#!/bin/bash
mkdir -p /var/homesnap
lvcreate -s -L 2G -n homesnap /dev/vg1/home
mount -t ext4 /dev/vg1/homesnap /var/homesnap
tar czf /backup/mybackup.tgz /var/homesnap
lvdisplay /dev/vg1/homesnap > /tmp/backup.log
umount /var/homesnap
lvremove -f /dev/vg1/homesnap
```

Команда `lvdisplay` применяется для того, чтобы контролировать, правильно ли вы определили размеры буфера памяти для мгновенного снимка LVM.

```
root# cat /var/homesnap
   ...
   COW-table size        2.00 GB
   Allocated to snapshot 5.23%
```

Итак, в предыдущем примере было занято всего 5 % буфера памяти. Разумеется, нет никакой гарантии, что и в следующий раз будет так же. Может быть, какой-то пользователь или процесс сделает большие изменения в каталоге `/home`.

Считывание логического тома в виде образа

Если вы используете логический том для того, чтобы сохранять там информацию с носителя данных виртуальной машины, то можно использовать мгновенные снимки LVM для надежного копирования логического тома в сжатый файл образа. Следующий сценарий сначала переименовывает резервную копию `image.lzo` в `old-image.lzo`, если она уже имеется. После этого он создает мгновенный снимок `snap` логического тома `/dev/vg1/lv1`, где содержится виртуальный носитель данных.

После этого образ считывается с помощью `cat` и сжимается посредством `lzop`. Благодаря `ionice -c 3` выполнение этой команды, значительно нагружающей механизмы процессора и ввода-вывода, происходит сравнительно легко, без замедления других процессов.

Затем сжатый образ загружается на сервер резервных копий с помощью `curl`. Параметр `--limit-rate` ограничивает при этом скорость передачи. Так мы гарантируем, что сетевые мощности сервера не будут полностью потрачены на обслуживание сценария резервного копирования.

```
#!/bin/bash
mv /backup/image.lzo /backup/old-image.lzo
lvcreate -s -L 2G -n snap /dev/vg1/lv1
ionice -c 3 cat /dev/vg1/snap | lzop -c > /backup/image.lzo
lvremove -f /dev/vg1/snap
# Загрузка образа по FTP на другой сервер резервных копий
pw=u12345:2n34jkj546wqdsr
ftp=u12345.backup-server.de
curl --limit-rate 8m -T /backup/image.lzo -u $pw ftp://$ftp/image.lzo
```

Tartarus

Не стоит изобретать велосипед, если в Интернете уже есть готовые сценарии для резервного копирования. Если поискать в Интернете по запросу `linux backup script`, вы получите массу результатов. Самое сложное в данном случае — отделить пшеницу от плевел. Многие сценарии ориентированы на выполнение довольно узких задач и не отличаются универсальностью. Но, к счастью, есть приятное исключение. Мне очень помогает в работе bash-сценарий Tartarus. Его код составляет около 600 строк, и в нем содержатся следующие функции:

- архивация в формате TAR;
- сохранение резервной копии в локальной файловой системе, на FTP-сервере или с помощью определяемой вами команды (например, `ssh`);
- инкрементные резервные копии;
- мгновенные снимки LVM;
- шифрование с помощью GPG.

26 Брандмауэры

В этой главе речь пойдет о работе в сети и различных механизмах защиты. Для этого мы познакомимся с некоторыми элементарными инструментами, помогающими анализировать актуальное состояние сети (например, находить открытые порты). Кроме того, в разделе о библиотеке TCP-Wrapper показано, как ограничить доступ к определенным сетевым службам с помощью простых правил.

26.1. Основы работы в сети и анализ сети

Приступая к обеспечению безопасности компьютера, нужно иметь представление о том, как работают сетевые службы, какие службы действуют в настоящее время, какие порты открыты и т. д. В этом разделе мы рассмотрим основы работы с TCP/IP и некоторые программы, позволяющие анализировать текущее состояние сети, например составлять список всех действующих сетевых соединений.

Прежде всего рассмотрим табл. 26.1, в которой в обобщенном виде представлены важнейшие сокращения.

Таблица 26.1. Важнейшие сокращения сетевых терминов

Сокращение	Значение
DNS	Служба доменных имен
HTTP	Протокол передачи гипертекста
ICMP	Протокол управления сообщениями в Интернете
IP	Интернет-протокол
NFS	Сетевая файловая система
TCP	Протокол управления передачей
UDP	Протокол пользовательских датаграмм

Интернет-протокол. Практически все распространенные сетевые службы базируются на IP-пакетах. Если, например, интернет-пользователь хочет обратиться к вашему компьютеру через FTP, то компьютер запускает FTP-клиент. Этот клиент посылает на ваш компьютер специальные пакеты. Если на вашем компьютере установлен FTP-сервер, он принимает эти IP-пакеты и реагирует на запрос, пересылая свои IP-пакеты клиенту.

Кроме самих данных, в IP-пакетах содержатся (в том числе) еще четыре важных фрагмента информации: IP-адрес отправителя, порт отправителя, адрес назначения и порт получателя. Благодаря этим данным становится известно, откуда приходит пакет и куда он должен быть направлен.

IP-адреса и порты. Вы уже понимаете, зачем нужен IP-адрес (см. главу 18). IP-порты применяются для идентификации различных служб. Например, для запроса веб-документа обычно используется порт 80. Номера портов — это 16-битные числа. Порты вплоть до 1024 считаются привилегированными и зарезервированы для серверных служб (например, для HTTP-сервера). Остальные порты могут использоваться и клиентами, но и среди них есть несколько номеров, которые не должны применяться клиентом, так как в свою очередь зарезервированы для выполнения определенных целей.

Для многих IP-номеров портов в `/etc/services` заданы псевдонимы. В табл. 26.2 перечислены важнейшие номера портов, а также имена, под которыми они обычно используются (если такие имена есть), и краткое объяснение.

Таблица 26.2. Важнейшие IP-порты

Название	Порт	Функция
ftp	20,21	FTP
ssh	22	SSH
telnet	23	Telnet
smtp	25	Электронная почта
domain	53	DNS
Bootps и bootpc	67, 68	DHCP
http	80	Сеть
kerberos	88	Kerberos
pop3	110	Электронная почта
portmap	111	Portmap (для NFS)
ntp	123	Время (сетевой протокол синхронизации времени)
netbios-ns	137	Служба имен Microsoft/NetBIOS
netbios-dgm	138	Служба датаграмм Microsoft/NetBIOS
netbios-ssn	139	Служба доступа к файлам Microsoft (SMB, Samba)
imap	143	Электронная почта
ldap	389	LDAP
	427	Файловый протокол Apple (AFP)
https	443	Сеть (зашифрованный)
microsoft-ds	445	Файловая система CIFS (SMB, Samba)
printer	515	Печать с использованием LPD/LPR
	548	Файловый протокол Apple (AFP)
ipp	631	Печать с использованием IPP/CUPS
rmi	1099	Удаленный вызов методов (Java)
	1433	Microsoft SQL Server
pptp	1723	PPTP/VPN
nfs	2049	NFS

Продолжение ⇩

Таблица 26.2 (продолжение)

Название	Порт	Функция
	3128	Squid (сетевые прокси)
mysql	3306	Сервер базы данных MySQL
	5353	Конфигурация сети с помощью Zeroconf/Bonjour
	5999-6003	X-дисплей
	9100	Сетевой принтер HP-JetDirect

IP-протоколы. Существуют различные протоколы для работы с IP-пакетами: большинство интернет-служб используют TCP. Этот протокол требует подтверждения о получении пакета. Но бывают и протоколы, которым такое подтверждение не нужно. К их числу относится, например, ICMP (применяется программой `ping`) и UDP (используется DNS и NFS).

Фильтр IP-пакетов. IP-пакеты могут создаваться локальными программами или приходить на компьютер извне — через сетевой или PPP-интерфейс. Ядро решает, как поступить с пакетами. Упрощенно говоря, ядро может либо отбросить данные пакеты, либо переадресовать работающим программам или другим интерфейсам. При этом описанные выше характеристики пакетов могут использоваться в качестве критериев для принятия решений. Чтобы применить такой фильтр пакетов на практике, необходимо сообщить ядру, как оно должно поступать с различными IP-пакетами. Для этого в ядре, начиная с версии 2.4, используется команда `iptables` (см. раздел 26.5).

Определение активных сетевых портов. Принцип работы большинства сетевых служб заключается в том, что эти службы наблюдают за определенным портом. Если на этот порт приходят IP-пакеты, то конкретная служба занимается обработкой пришедшей информации и отвечает на нее. Пакеты, которые были присланы на ненаблюдаемые порты, просто игнорируются и поэтому не представляют опасности.

Чтобы оценить степень опасности, которой подвергается компьютер, нужно получить список всех наблюдаемых портов (справедливо и обратное — злоумышленник, атакующий компьютер, в первую очередь попытается узнать номера активных портов).

Netstat. При определении сетевой активности локального компьютера очень помогает команда `netstat`. В зависимости от того, с какими параметрами команда вызывается, она выдает массу различной информации.

В первом примере (сервер mars) отображаются все активные соединения (`established`) или наблюдаемые порты (`LISTEN`). Кратко опишу параметры: `a` — отображает неактивные порты; `tu` — выводит только ту информацию, которая касается портов TCP и UDP; `pe` — дополнительно отображает номер процесса и учетную запись, под которой он выполняется. Ради экономии места вывод сокращен.

```
root# netstat -atupe
Active Internet connections (servers and established)
Proto Local Address          Foreign Addr  State   User    PID/Prog name
tcp   *:nfs                  *:*           LISTEN  root    -
tcp   *:54980                *:*           LISTEN  root    -
```

```
tcp   *:ldap                *:*          LISTEN  root    5842/slapd
tcp   *:3142                *:*          LISTEN  root    5904/perl
tcp   localhost:mysql       *:*          LISTEN  mysql   5785/mysqld
...
tcp6  [::]:ssh              [::]:*       LISTEN  root    5559/sshd
tcp6  mars.sol:ssh          merkur.so...  ESTAB  root    7729/0
udp   *:nfs                 *:*                  root    -
udp   mars.local:netbios-ns *:*                  root    6231/nmbd
udp   mars.sol:netbios-ns   *:*                  root    6231/nmbd
udp   *:netbios-ns          *:*                  root    6231/nmbd
udp   mars.local:netbios-dgm *:*                 root    6231/nmbd
udp   mars.sol:netbios-dgm  *:*                  root    6231/nmbd
udp   *:netbios-dgm         *:*                  root    6231/nmbd
udp   *:domain              *:*                  root    5537/dnsmasq
udp   *:55350               *:*                  avahi   5604/avahi-...
...
```

Кратко опишу этот вывод: на тестовом компьютере работают серверы Samba, NFS, Kerberos, LDAP, Dnsmasq, CUPS, MySQL и SSH. Если компьютер напрямую подключен к Интернету (без брандмауэра), то любой потенциальный агрессор злорадно потирает руки. Существует множество программ, безопасность которых несовершенна.

Следующая команда возвращает список активных TCP- и UDP-соединений вместе с именами пользователей и названиями процессов:

```
root# netstat -tuep
Active Internet connections (w/o servers)
Proto Local Address   Foreign Address  State       User     PID/Program name
tcp   localhost:57450 localhost:ldap   ESTABLISHED root     6233/smbd
tcp   localhost:ldap  localhost:57450  ESTABLISHED openldap 5842/slapd
tcp6  mars.sol:ssh    merkur.sol:45368 ESTABLISHED root     7729/0
```

Lsof. Если требуется узнать, какие программы используют порты TCP и UDP, вам пригодится команда `lsof`. В форме `derFormlsof -i [протокол]@[хост-имя][:порт]` команда возвращает список процессов, использующих указанные сетевые ресурсы. Две следующие команды отображают все процессы, применяющие протокол UDP или порт 22:

```
root# lsof -i udp
ntpd       3696  ntp    16u   IPv4    9026  UDP *:ntp
ntpd       3696  ntp    17u   IPv6    9028  UDP *:ntp
ntpd       3696  ntp    18u   IPv6    9031  UDP ip6-localhost:ntp
portmap    4745  daemon  3u   IPv4   12931  UDP *:sunrpc
rpc.statd  4764  statd   5u   IPv4   12962  UDP *:700
rpc.statd  4764  statd   7u   IPv4   12970  UDP *:39146
...
root# lsof -i :22
COMMAND  PID USER FD TYPE DEVICE  SIZE  NODE  NAME
sshd    5559 root 3u IPv6        14097   TCP  *:ssh (LISTEN)
sshd    7729 root 3r IPv6        33146   TCP  mars.sol:ssh->merkur.sol:45368
                                                        (ESTABLISHED)
```

Nmap. Команды netstat и lsof могут выполняться только на локальном компьютере, то есть у злоумышленников не получится ими воспользоваться. Но враг может применить так называемый сканер портов. Такие программы рассылают пакеты на важнейшие порты компьютера и на основании ответа определяют, какие службы (и какие именно версии этих служб) работают на компьютере. Представленная здесь команда nmap — известнейший, но далеко не единственный такой сканер. В большинстве дистрибутивов она устанавливается при первом запуске.

В следующих строках показано, какие результаты nmap возвращает для сервера mars. Команда nmap выполнялась на другом компьютере в той же локальной сети. Вывод сокращен ради экономии места.

```
root# nmap -v -A mars
Starting Nmap 4.62 ( http://nmap.org ) at 2009-03-20 09:43 CET
Initiating ARP Ping Scan at 09:43
Scanning 192.168.0.1 [1 port]
...
Discovered open port 53/tcp on 192.168.0.1
Discovered open port 21/tcp on 192.168.0.1
...
Completed SYN Stealth Scan at 09:43, 0.29s elapsed (1715 total ports)
Initiating Service scan at 09:43
Scanning 9 services on mars.sol (192.168.0.1)
...
Host mars.sol (192.168.0.1) appears to be up ... good.
Interesting ports on mars.sol (192.168.0.1):
Not shown: 1706 closed ports
PORT    STATE   SERVICE         VERSION
21/tcp  open    ftp vsftpd      2.0.6
22/tcp  open    ssh OpenSSH     4.7p1 Debian 8ubuntu1.2 (protocol 2.0)
53/tcp  open    domain dnsmasq  2.41
111/tcp open    rpcbind
139/tcp open    netbios-ssn     Samba smbd 3.X (workgroup: SOL)
389/tcp open    ldap OpenLDAP   2.2.X
445/tcp open    netbios-ssn     Samba smbd 3.X (workgroup: SOL)
749/tcp open    rpcbind
2049/tcp open   rpcbind
MAC Address: 00:14:6C:8E:D9:71 (Netgear)
Device type: general purpose
Running: Linux 2.6.X
...
```

Для nmap также существует графический пользовательский интерфейс, который в зависимости от дистрибутива может находиться в пакете zenmap или nmap-frontend.

ВНИМАНИЕ

Сканирование портов воспринимается многими администраторами как покушение на взлом. Никогда не посылайте запросы с таких программ, как nmap, на незнакомые компьютеры! Однако nmap — это удобный и практичный инструмент, позволяющий обнаружить бреши в защите собственной сети.

26.2. Основы защиты сетевых служб

В предыдущем разделе было показано, как можно быстро получить информацию о выполняемых в данный момент сетевых службах. Следующий шаг заключается в том, чтобы как можно надежнее защитить эти службы.

- Деинсталлируйте все сетевые службы, которые вам не нужны. Службы, которые не установлены, не функционируют и поэтому совершенно безопасны.
- Часто при работе с самыми нужными сетевыми службами бывает достаточно лишь предоставить доступ к службе всего для нескольких определенных клиентов (которые, в частности, находятся в локальной сети). Например, практически исключен случай, в котором вам пришлось бы предоставлять доступ к службам сервера печати в Интернете.

  Что касается Apache, Samba, MySQL и многих других крупных служб, то меры защиты нужно предпринимать в соответствующем конфигурационном файле. К счастью, есть несколько сетевых служб, которые для управления доступом обращаются к библиотеке TCP-Wrapper. Эта библиотека позволяет осуществлять централизованную конфигурацию.
- Необходимые сетевые службы должны выполняться с минимальным набором прав. Их выполнением занимается система Init-V, присутствующая в вашем дистрибутиве. Если это возможно и целесообразно, запустите службы без прав администратора с учетной записи, созданной специально для этих целей, или в среде `chroot`, которая не позволит обращаться к файлам из-за пределов каталога chroot.
- В качестве дополнительного уровня защиты рекомендуется использовать специальный брандмауэр, который предназначен для фильтрации пакетов и который в соответствии с определенными правилами блокирует пакеты, приходящие из Интернета на адреса различных служб (см. раздел 26.3).
- Ни одна программа не защищена от ошибок. Программные ошибки позволяют потенциальным агрессорам завершать ваши программы с помощью отправки на компьютер специальных пакетов, а в особо тяжелых случаях даже выполнять на вашем компьютере свои команды. Чтобы свести к минимуму связанный с этим риск, ядро может наблюдать за выполнением программ на основании заранее заданных правил. Такой метод называется *мандатным управлением доступом* (Mandatory Access Control, MAC). В Linux для реализации этой функции используются два метода: SELinux и AppArmor (см. главу 29).

Обновления, журналирование. Невозможно обеспечить достаточную безопасность действующей конфигурации компьютера за один раз — только с помощью регулярных обновлений вы сможете поддерживать ваше ПО в актуальном состоянии. Рекомендуется регулярно просматривать файлы регистрации вашего компьютера.

Библиотека TCP-Wrapper

На сервере локальной сети обычно не следует делать все сетевые службы глобально доступными. Вполне достаточно обеспечить доступ к службам в рамках

локальной сети. Некоторые сетевые службы обращаются для обеспечения такой базовой защиты к библиотеке TCP-Wrapper. К подобным службам относятся, в частности, SSH- и NFS-сервер. Службы, которые запускаются *демоном интернет-сервисов*, также применяют библиотеку TCP-Wrapper (см. раздел 26.2).

Файлы /etc/hosts.allow и hosts.deny

Файлы `/etc/hosts.allow` и `/etc/hosts.deny` определяют, с каких компьютеров какими службами можно управлять. Настройки действуют только для тех сетевых служб, которые применяют для контроля доступа библиотеку TCP-Wrapper или команду `tcpd`. По умолчанию оба файла пусты, то есть никакие ограничения не действуют.

Сначала библиотека TCP-Wrapper интерпретирует `hosts.allow`: если в этом файле прямо указано, что доступ разрешен, то осуществляется контроль. В противном случае библиотека интерпретирует `hosts.deny`: если в этом файле доступ запрещен, то клиент получает сообщение об ошибке. Внимание: во всех случаях, для которых отсутствуют правила и в `/etc/hosts.allow`, и в `/etc/hosts.deny`, доступ разрешается!

Чтобы обеспечить максимально безопасную конфигурацию, нужно запретить запуск любых сетевых служб — это делается с помощью команды `all:all` в `/etc/hosts.deny`. Благодаря команде `spawn` любая попытка запустить сетевую службу будет протоколироваться в файле `/var/log/deny.log`.

Теперь из файла `/var/log/deny.log` вы сможете узнать, кто и когда пытается использовать сетевые службы вашего компьютера (запись в этом файле — необязательно сигнал об атаке; вполне возможно, что кто-то просто ошибся при указании IP-адреса).

```
# /etc/hosts.deny
# По умолчанию запретить все службы, любые попытки соединения
# протоколировать
ALL : ALL : spawn (echo Attempt from %h %a to %d at $(date) \
                   >> /var/log/deny.log)
```

Следующий шаг — разрешить в файле `/etc/hosts.allow` использование определенных служб. Приведенная далее в качестве примера конфигурация позволяет:

- получать с локального компьютера (localhost) доступ к любым службам;
- обмениваться информацией по `ssh` с любого компьютера, находящегося в сети (это касается любых компьютеров, подключенных к Интернету);
- использовать NFS и SWAT в локальной сети.

В примере предполагается, что сервер доступен под именами mars и mars.sol, что локальная сеть работает в адресном пространстве 192.168.0.* и что все компьютеры используют домен *.sol. Я просто даю шаблон, по которому вы можете активизировать для работы в локальной сети или в глобальном масштабе и другие сетевые службы, которые ранее отключили:

```
# /etc/hosts.allow
# Разрешить выполнение отдельных служб
ALL     : localhost mars mars.sol          : ALLOW
```

```
Sshd    : ALL                                : ALLOW
portmap : 192.168.0.0/255.255.255.0 *.sol  : ALLOW
mountd  : 192.168.0.0/255.255.255.0 *.sol  : ALLOW
swat    : 192.168.0.0/255.255.255.0 *.sol  : ALLOW
# Только в SUSE
cupsd   : 192.168.0.0/255.255.255.0 *.sol  : ALLOW
```

Синтаксис, применяемый в `hosts.allow` или `hosts.deny`, на основании данных примеров уже должен быть понятен. Каждая запись состоит из двух частей, разделяемых двоеточием. В первой части указывается служба, во второй — IP-адрес или сетевое имя, в третьей части — итоговое действие. В `man 5 hosts_access` содержится более подробное описание синтаксиса.

Обеспечение поддержки библиотеки TCP-Wrapper

Строка `cupsd` требуется в старых версиях SUSE (до 11.2 включительно), потому что сервер печати CUPS в этом дистрибутиве компилируется с поддержкой библиотеки TCP-Wrapper (чего не скажешь о большинстве других дистрибутивов). Но при этом необходимо учитывать, что доступ к сетевому принтеру дополнительно управляется CUPS-специфичным файлом `/etc/cups/cups.conf`.

С помощью команды `ldd` вы легко можете сами определить, использует ли та или иная программа библиотеку TCP-Wrapper (`libwrap`). Результаты для `cupsd` и `sshd` в openSUSE и Ubuntu выглядят так:

```
user$ ldd /usr/sbin/cupsd | grep wrap       (openSUSE 11.3 и выше)
user$ ldd /usr/sbin/cupsd | grep wrap       (openSUSE 11.2 и выше)
   libwrap.so.0 => /lib64/libwrap.so.0 (0x00007f1f3fece000)
user$ ldd /usr/sbin/cupsd | grep wrap       (Ubuntu)
user$ ldd /usr/sbin/sshd | grep wrap
   libwrap.so.0 => /lib/libwrap.so.0 (0x00007f1a5f7f0000)
```

Запуск сетевых служб без прав администратора

Чтобы такие программы, как Apache и MySQL, без проблем выполняли за вас вашу работу, нет необходимости запускать их с правами администратора. В большинстве дистрибутивов для таких служб предусмотрены специальные учетные записи, названия которых различаются от дистрибутива к дистрибутиву. Например, в Ubuntu Apache выполняется под учетной записью `www-data` и поэтому может обращаться лишь к тем файлам, которые доступны для чтения с этой учетной записи. Вы можете убедиться в этом с помощью команды `ps axu` (но один экземпляр Apache все же должен работать с правами администратора; он отвечает лишь за запуск других экземпляров Apache и не выполняет никаких других задач).

```
root# ps axu | grep apache2
root ...     /usr/sbin/apache2 -k start
www-data ... /usr/sbin/apache2 -k start
www-data ... /usr/sbin/apache2 -k start
...
```

Поскольку сценарии систем Init, как правило, выполняются с правами администратора, для запуска сетевого демона с другой учетной записи требуется специальный

механизм. В простейшем случае процесс необходимо запустить в форме `su` *`имя учетной записи`* `-c` *`демон`*. Но для большинства сетевых процессов все же предусмотрены более тонкие механизмы, в ходе реализации которых программа инициализируется с правами администратора и только потом переходит к работе с учетной записью, для которой определено меньшее количество прав. В некоторых программах, например в `syslog`, предусмотрен специальный параметр, позволяющий указать нужную учетную запись. У Apache, MySQL и некоторых других серверных служб, которые запускаются в нескольких экземплярах, управляющий процесс сохраняет за собой права администратора. Но обычно такой процесс выполняет нетипичные задачи (как правило, запуск и остановку других экземпляров).

Запуск сетевых служб в среде chroot

Команда `chroot` *`каталог команда`* запускает указанную команду, причем *`каталог`* используется в качестве корневого каталога. Команда может обращаться только к тем файлам, которые находятся в пределах этого каталога. Чтобы гарантировать, что программа не сможет «выбраться» из своей «клетки», ее нужно выполнять с учетной записи, имеющей ограниченные права (то есть учетная запись root не подходит).

Правда, на практике сетевые службы запускаются не с помощью `chroot`, а посредством специального параметра, предназначенного для указания `chroot`-каталога. В этом каталоге должны находиться все необходимые библиотеки, конфигурационные файлы и т. д. Все эти файлы скопирует в нужный каталог перед запуском сценарий Init-V.

В SUSE по умолчанию DHCP-сервер и сервер имен запускаются в средах `chroot`. В качестве DHCP-сервера в данном случае используется специальная обновленная версия `dhcpd`, в которой `chroot`-каталог можно задавать с помощью дополнительного параметра `-chroot`. Что касается сервера имен, то для указания `chroot`-каталога применяется параметр `-t`.

Если за сетевой службой наблюдает SELinux или AppArmor и правила обеспечения безопасности сформулированы правильно, то можно обойтись и без применения среды `chroot`, а если она и используется, то никак не способствует повышению безопасности. В Fedora и Red Hat `chroot`-каталоги по умолчанию не применяются, а система полностью полагается на соблюдение правил SELinux.

26.3. Брандмауэры: общая информация

Термин «брандмауэр» у всех на устах, но общепринятого определения этого феномена не существует. Функции брандмауэра могут выполняться оборудованием: в таком случае под брандмауэром обычно понимается компьютер, стоящий на стыке локальной сети и Интернета. Многие ADSL-роутеры могут иметь простейшие функции брандмауэров.

Нередко брандмауэром может быть и программный пакет, установленный на компьютере и при условии правильной конфигурации повышающий безопасность компьютера. Во многих дистрибутивах содержатся многофункциональные инструменты, предназначенные для настройки конфигурации брандмауэра.

В этой книге под брандмауэром понимается совокупность методов, повышающих надежность обмена информацией, проходящей по TCP/IP через фильтр пакетов. Такой фильтр анализирует все сетевые пакеты, приходящие на компьютер, а также пакеты, которые уходят с компьютера в сеть. В зависимости от того, все ли правила соблюдаются, пакеты могут быть пропущены или заблокированы. Конфигурация подобного фильтра пакетов подробно рассмотрена в следующем разделе. Но сначала разберемся с терминологией.

Брандмауэры для частных ПК

Сегодня большинство частных ПК постоянно подключены к Интернету, доступны по фиксированному IP-адресу, назначаемому провайдером, а значит, подвергаются опасности.

Если, к примеру, на компьютере действует SSH-сервер, то злоумышленник может попытаться войти в сеть через этот сервер. Для этого агрессоры используют сценарии, автоматически подбирающие логины, просто подставляя слова из словаря. Таким образом, хороший пароль дорогого стоит! Другая опасность таится во WLAN: на настоящий момент хорошо защищенными можно считать только те сети WLAN, в которых применяется механизм WPA2, и то лишь при условии, что используемый пароль является достаточно длинным и сложным.

Вы можете возразить, что атака на ваш компьютер не имеет смысла, ведь находящиеся на нем данные вряд ли кого-то заинтересуют. Может быть, и так. Но не каждая атака предпринимается с целью выведать данные, а потом манипулировать ими. Часто злоумышленник хочет установить у вас на компьютере маленькую программу, которой позже сможет воспользоваться. Жертвами подобных атак становятся миллионы компьютеров с Windows, которыми могут удаленно управлять злоумышленники.

Брандмауэры для локальных сетей

Обычно корпоративные локальные сети больше нуждаются в обеспечении безопасности, чем домашние ПК. Одновременно создаются лучшие условия для построения нужной инфраструктуры. На практике в фирменной локальной сети за выход в Интернет и обеспечение безопасности часто отвечает отдельный компьютер. Все остальные сетевые службы работают на других компьютерах. Эта концепция изображена на рис. 26.1.

В очень небольших сетях функции брандмауэра и сетевого сервера может выполнять один компьютер. Но такой подход не является оптимальным, так как на этом компьютере придется запустить и эксплуатировать множество сетевых служб, которые могут быть использованы для нанесения вреда сети.

В очень больших сетях часто бывает не один, а даже два брандмауэра. Первый служит лишь для обеспечения базовой безопасности, но пропускает такие интернет-протоколы, как HTTP или FTP. Сетевое пространство в таком случае называется *демилитаризованной зоной* (Demilitarized Zone, DMZ). Этот термин означает, что внутри сети лишь ограниченные меры безопасности. Как правило, в этой зоне располагается веб-сервер, а также другие сетевые серверы, которые должны быть общедоступны (то есть могут быть найдены через Интернет).

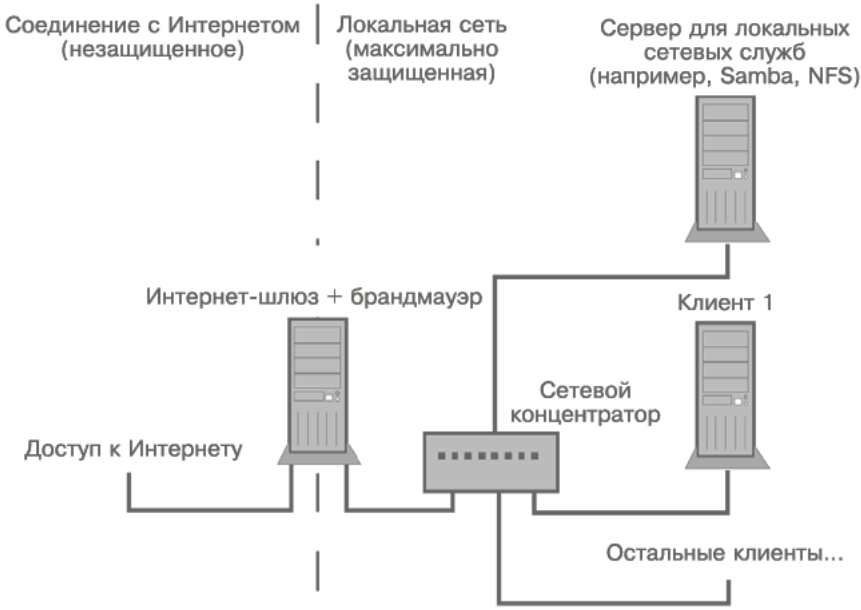


Рис. 26.1. Брандмауэр для локальной сети

Демилитаризованная зона отделяется от оставшейся части локальной сети вторым брандмауэром. Уже за ним располагаются все другие службы, отвечающие за работу локальной сети и абсолютно недоступные извне. Однако конфигурация многоступенчатого брандмауэра — очень обширная тема, выходящая за рамки этой книги. Руководства по конфигурации таких барьеров даются в специальной литературе.

Сетевой фильтр

Внутри ядра обработкой правил брандмауэра занимается система, называемая *сетевым фильтром*. На рис. 26.2 в очень упрощенном виде показано, какими путями IP-пакеты могут передвигаться в системе фильтрации пакетов. Более подробная схема дается по следующему адресу: http://open-source.arkoon.net/kernel/kernel_net.png.

В следующем списке очень кратко описаны состояния IP-пакета в ядре.

- **Routing (Маршрутизация).** Основываясь на информации об IP и адресе порта, ядро решает, должен пакет обрабатываться на локальном компьютере или его следует передать через сетевой интерфейс на другой компьютер (который может находиться как в локальной сети, так и в Интернете).
- **Filter Input (Входной фильтр).** На основании определенных правил проверяется, следует ли обработать пришедший пакет с помощью локальных программ (например, сетевых демонов).

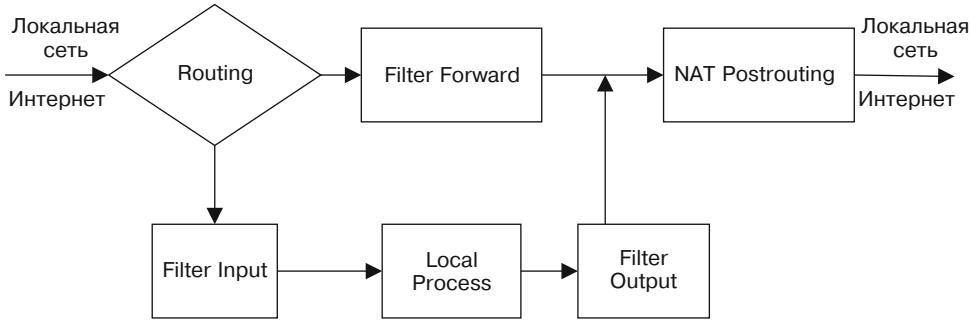


Рис. 26.2. Упрощенное представление системы iptables/netfilter

- **Local Process (Локальный процесс).** В этом окне символически изображены все программы, обрабатывающие IP-пакеты на локальном компьютере либо создающие IP-пакеты (то есть это все сетевые службы, например FTPD, HTTPD).
- **Filter Output (Выходной фильтр).** На основании отдельных правил определяется, может ли IP-пакет снова покинуть ядро.
- **Filter Forward (Фильтр переадресации).** Этот фильтр определяет, какие пакеты, которые следует только переадресовать (но не обрабатывать), могут пройти через ядро.
- **NAT Postrouting.** Если компьютер должен предоставлять другим компьютерам доступ в Интернет путем маскарадинга, этот механизм выполняет нужные операции с IP-пакетами других компьютеров.

Фильтр пакетов на рис. 26.2 соединяется только с окнами Filter Input, Filter Output, Filter Forward и, возможно, NAT Postrouting. В остальных частях изображения описаны сетевые функции ядра или обычные сетевые функции, работающие в локальной системе и никак не связанные с работой фильтра пакетов.

ВНИМАНИЕ

На многих схематических изображениях брандмауэра слева рисуется полный опасностей Интернет, в центре — брандмауэр, а справа — надежная и безопасная локальная сеть. Так вот, на рис. 26.2 все построено иначе! Пакеты, которые слева идут в компьютер, происходят как из локальной сети, так и из Интернета. То же касается и пакетов, которые справа проходят через брандмауэр.

Действия. За переадресацию пакетов отвечает ядро — независимо от того, приходят они с сетевого интерфейса или создаются на компьютере одной из локальных программ. Ядро может совершать на различных уровнях фильтрующей системы три разных действия.

- **Deny** — переадресация пакетов отклоняется без запроса о подтверждении (можно сказать, что при этом пакет удаляется и больше не существует).
- **Reject** — переадресация пакетов отклоняется с запросом о подтверждении. С пакетом происходит то же, что и в первом случае, но отправитель с другим ICMP-пакетом получает уведомление о том, что его пакет не был принят.
- **Accept** — пакет переадресовывается.

Таблицы. В принципе, сетевой фильтр построен так: каждый IP-пакет проходит через различные части ядра, где в определенных пунктах проверяется на основании установленных правил. При соответствии правилам пакет переадресовывается, в противном случае — удаляется или отправляется назад. Сетевой фильтр управляется тремя таблицами.

- **Таблица фильтра** — обычно в ней содержится вся система правил для отдельных пакетов (брандмауэр).
- **Таблица трансляции сетевых адресов** — действует лишь в том случае, если в ядре активизирована функция маскарадинга. Она обеспечивает возможность различных изменений адресов (трансляции сетевых адресов) для пакетов, приходящих в ядро извне либо покидающих ядро.
- **Таблица mangle** — также позволяет выполнять различные операции с IP-пакетами. Таблица служит для выполнения специальных задач и далее в книге рассматриваться не будет.

Цепочки правил (chains). В каждой из упомянутых таблиц предусмотрено несколько цепочек правил:

- таблица фильтра — Input, Forward, Output;
- таблица трансляции сетевых адресов — Prerouting, Output и Postrouting;
- таблица mangle — Prerouting и Output.

Из восьми этих цепочек правил на рис. 26.2 отображены только четыре самые важные.

ПРИМЕЧАНИЕ

Цепочки правил не зависят друг от друга, то есть существует две разные цепочки Prerouting и три разные цепочки Output. Однако в документации часто пишут, что речь идет просто о цепочке Output, не указывая, к какой именно таблице она относится. В таких случаях всегда имеются в виду цепочки таблицы-фильтра — далее мы увидим, что эта таблица является наиболее важной.

Такая же трактовка касается и команды iptables: там с помощью параметра -t можно указать желаемую таблицу. Если этого параметра нет, то команда будет автоматически применена к таблице-фильтру.

Когда идущий через ядро IP-пакет сталкивается с цепочкой правил, ядро по порядку проверяет его на соответствие всем правилам. Как только одно из определенных правил совпадает с характеристикой пакета, над пакетом выполняется предусмотренное на этот случай действие (например, пакет переадресовывается, удаляется, отправляется обратно и т. д.). Только если с характеристиками пакета не совпадает ни одно из указанных правил, выполняется действие, заданное в системе по умолчанию. В зависимости от конфигурации по умолчанию также может быть задано одно из трех действий: переадресовать, удалить, отправить обратно.

Исходное состояние. В исходном состоянии в ядре активна только таблица-фильтр с тремя цепочками Input, Forward, Output. Ни одна из этих цепочек не содержит дополнительных правил и для всех трех по умолчанию задано действие по переадресации.

Команда iptables. Итак, искусство создания фильтра пакетов заключается в том, чтобы определить для каждой релевантной цепочки фильтра действие, выполня-

емое по умолчанию, а также несколько правил. Для этого применяется команда `iptables`. Конкретный пример ее использования приведен в следующем разделе. Более подробно эта проблема, как обычно, освещена в Интернете:

- http://www.netfilter.org/;
- http://www.linuxguruz.org/iptables/;
- http://people.netfilter.org/rusty/unreliable-guides/.

Система nftables. Система сетевого фильтра входит в состав ядра с 2001 года. Со временем обнаружились определенные недостатки этой системы, и уже ведется работа над системой следующего поколения `nftables`. Однако пока еще рано судить, когда и в какой форме она заменит сетевой фильтр. Информацию вы можете найти на сайте http://lwn.net/Articles/324989/.

26.4. Создание брандмауэра (помощь в конфигурации)

В этом разделе мы научимся создавать фильтр пакетов с помощью сценария, состоящего из многочисленных команд `iptables`. Но для этого вы должны очень хорошо изучить материал. Для среднего пользователя Linux эти умения определенно не требуются. Поэтому производители многих дистрибутивов выполняют данную работу за пользователя. С помощью исключительно удобных конфигурационных инструментов пользователь может настроить простой брандмауэр едва ли не одним нажатием кнопки — частично уже на этапе установки. Сами по себе такие конфигурационные инструменты, конечно же, хороши. За, казалось бы, совсем простой конфигурацией может скрываться множество патентованных фильтров-пакетов. В результате защита обычно получается лучше, чем если смастерить брандмауэр самостоятельно.

Недостаток таких брандмауэров заключается в том, что они ведут себя, в сущности, как «черный ящик». Даже если принцип работы фильтра документирован, эта документация обычно оставляет желать лучшего. Вы не знаете, ни от чего он вас защищает, ни какие побочные эффекты имеет. Может случиться так, что начинающий пользователь Linux потратит несколько дней на выяснение того, почему же не работает печать документов по сети, пока не поймет, что все дело в брандмауэре. Но если после этого попытаться откорректировать фильтр пакетов, то, скорее всего, ничего не получится, потому что пользователь не имеет базовых знаний по работе с брандмауэрами. И вот ожидаемый результат: брандмауэр просто будет отключен!

Debian. По умолчанию в Debian брандмауэр не предусмотрен. В дистрибутиве отсутствуют и специальные инструменты для его конфигурации.

Fedora и Red Hat. Во время установки Fedora и Red Hat по умолчанию создается брандмауэр, блокирующий все приходящие извне попытки соединения. Для настройки запустите программу `system-config-firewall`.

Здесь можно определить отдельные службы (например, SSH) и отдельные сетевые интерфейсы (например, LAN) как надежные и не защищать систему от них.

Если компьютер работает в качестве шлюза, то можно задать и интерфейс для маскарадинга. Брандмауэр запускается системой Init:

- файл правил — `/etc/sysconfig/iptables`;
- сценарий Init-V — `/etc/init.d/iptables`;
- служебный файл Systemd — `/lib/systemd/system/iptables.service` (Fedora).

Fedora версии 17 и выше. По умолчанию описанная выше система работы с брандмауэром применяется и в Fedora 17. Но в отличие от более ранних версий существенно изменена внутренняя структура брандмауэра (новый пакет `firewalld`). В основе системы лежит работающий в фоновом режиме демон брандмауэра, позволяющий динамически вносить в брандмауэр изменения. (В традиционных системах брандмауэров после каждого изменения требуется перезапуск всего брандмауэра.)

В будущем для конфигурации FirewallD должна появиться программа с графическим интерфейсом `firewall-config`. Но на момент выпуска Fedora 17 эта программа была еще не готова. Предположительно она появится только в версии 18. До тех пор настройку придется осуществлять вручную с помощью команды `firewall-cmd` или в файлах каталога `/etc/firewalld`. Но из-за скудной документации это сложная задача. Общая информация об актуальном состоянии проекта FirewallD изложена здесь: http://fedoraproject.org/wiki/FirewallD.

SUSE. В дистрибутивах SUSE брандмауэр также создается по умолчанию, причем интерфейс для соединения с Интернетом автоматически относится к внешней зоне. Настройка происходит в модуле Безопасность ▸ Брандмауэр. При этом в отличие от общей организации диалоговых окон в YaST отдельные вкладки представлены в виде записей в левой части окна YaST (там, где располагается область справки). Если компьютер работает в качестве шлюза локальной сети, на вкладке Интерфейсы нужно отнести интерфейс LAN к внешней зоне, а на вкладке Маскарадинг установить одноименный флажок. Брандмауэр запускается процессом Init-V:

- файл правил — `/etc/sysconfig/SuSEfirewall`;
- сценарий Init-V — `/etc/init.d/SuSEfirewall2*`.

Ubuntu. По умолчанию в Ubuntu брандмауэр не создается, специальные инструменты с графическим интерфейсом для этого отсутствуют. Для такой конфигурации в Ubuntu присутствует команда `ufw` (Uncomplicated Firewall, несложный брандмауэр). Она позволяет определять правила работы брандмауэра с помощью сравнительно простого синтаксиса в отличие от `iptables`. Кроме того, в будущих версиях Ubuntu при установке сетевых служб будут попутно устанавливаться правила `ufw`, обеспечивающие защиту. В любом случае это еще отдаленная перспектива: до сих пор `ufw` распространена мало.

Если вы уже имеете опыт работы с фильтрами пакетов, то с помощью `ufw` быстро добьетесь нужного результата. Кратко расскажу о ее синтаксисе (подробнее вы можете прочитать в справке `man ufw`): `ufw enable` активизирует брандмауэр. В будущем брандмауэр будет активизироваться при каждом запуске компьютера. Команда `ufw default allow|deny` указывает, должны ли поступающие извне пакеты в принципе приниматься или отклоняться (как правило, действует `deny`). Дополнительно с помощью `ufw allow|deny` *n* или `ufw allow|deny` *служба* вы определяете правила, кото-

рые будут действовать для конкретных IP-портов или протоколов. Команда `ufw status` дает информацию об актуальном состоянии брандмауэра.

```
user$ sudo -s
root# ufw enable
root# ufw allow ssh
root# ufw status
Firewall loaded

To          Action      From
--          ------      ----
22:tcp      ALLOW       Anywhere
22:udp      ALLOW       Anywhere
```

Собственные правила сохраняются в `/lib/ufw/user.rules` и `user6.rules` (для IPv6). Кроме того, при работе `ufw` учитываются файлы правил из каталога `/etc/ufw/`. Более подробная информация и примеры работы с `ufw` приводятся на сайтах https://help.ubuntu.com/12.04/serverguide/firewall.html и https://wiki.ubuntu.com/UncomplicatedFirewall.

`Gufw` — это программа с графическим интерфейсом для работы с `ufw`. В современных версиях Ubuntu она предоставляется в виде `universe`-пакета (то есть без официальной поддержки). Подробнее об этой программе рассказано на сайте https://help.ubuntu.com/community/Gufw.

Firestarter. Это популярная, не зависящая от конкретного дистрибутива программа для конфигурации брандмауэров. При первом запуске открывается ассистент для создания базовой конфигурации. Сначала выбирается интерфейс, через который все компьютеры соединяются с Интернетом. При установке флажка Запуск с выбором брандмауэра брандмауэр автоматически активизируется, как только появляется соединение с Интернетом. Если ваш компьютер обращается за сетевыми параметрами на DHCP-сервер, вам также потребуется установить флажок Получать IP-адрес от DHCP-сервера.

На втором этапе вы можете конфигурировать компьютер как шлюз. При необходимости Firestarter одновременно может настроить DHCP-сервер `dhcpd`, но для этого предварительно требуется установить специальный пакет. Завершите работу ассистента, нажав кнопку Сохранить, но не Закончить, иначе сделанные настройки будут сброшены.

Брандмауэр сразу же становится активен и в дальнейшем автоматически запускается при включении компьютера или при установлении соединения с Интернетом. Теперь ни один компьютер, находящийся в локальной сети или в Интернете, не может начать обмен информацией с вашим компьютером. Часто эта настройка оказывается излишне жесткой. Потребуются дополнительные правила, которые позволят другим компьютерам обмениваться информацией с вашим.

Чтобы задать новые правила, выполните Система ▸ Управление системой ▸ Быстрый запуск, откройте вкладку Безопасность, выберите Правила для поступающего трафика и щелкните кнопкой мыши на области списка Разрешать подключения или Разрешать службу. Только теперь станет активной кнопка Добавить указание, которая открывает конфигурационное диалоговое окно. Новое правило активизируется нажатием кнопки Применить указание.

Более подробная информация о конфигурации и применении Firestarter дается на следующем сайте: http://www.fs-security.com/.

Ссылки. Кроме упомянутых выше программ есть еще множество инструментов, которые помогут вам настроить конфигурацию брандмауэра. Существуют инструменты с графическим пользовательским интерфейсом и без него. Посмотрите следующие сайты:

- http://www.fwbuilder.org/;
- http://www.simonzone.com/software/guarddog/;
- http://firehol.sourceforge.net/;
- http://www.shorewall.net/.

Количество правил фильтра. Команда `iptables -L | wc -l` позволяет оценить, какое количество правил используется при работе активного в данный момент брандмауэра. Итоговое количество позволяет судить о том, насколько сложен брандмауэр, но ничего не говорит о степени его надежности! Лучше всего полностью запретить весь сетевой трафик, а для этого обычно достаточно одного-двух правил.

26.5. Как самостоятельно построить брандмауэр с помощью iptables

В данном разделе будет рассмотрено программирование собственного брандмауэра для фильтрации пакетов для шлюза, то есть для компьютера, предоставляющего всем остальным компьютерам локальной сети доступ в Интернет. На рис. 26.1 показано исходное положение. Задача брандмауэра заключается в том, чтобы полностью заблокировать порты, представляющие опасность, и разрешать обмен информацией через другие порты только при условии, что такой обмен информацией инициирован изнутри системы (то есть из локальной сети). Наконец, сценарий брандмауэра также занимается активацией функций маскарадинга.

Пример был выполнен и протестирован в Ubuntu. Если вы работаете с другим дистрибутивом, то вам обязательно нужно сначала деактивировать брандмауэр, работающий в вашем дистрибутиве! Кроме того, нужно откорректировать сценарий Init в соответствии с правилами, действующими в вашем дистрибутиве.

Брандмауэр состоит из двух сценарных файлов `myfirewall-start` и `myfirewall-stop`. Базовые настройки брандмауэра сохраняются в конфигурационном файле `myfirewall`.

```
/etc/myfirewall/myfirewall-start   (Запуск брандмауэра)
/etc/myfirewall/myfirewall-stop    (Остановка брандмауэра)
/etc/default/myfirewall            (Базовые настройки)
```

Базовая конфигурация (myfirewall)

Переменная `MFW_ACTIVE` в `/etc/default/myfirewall` определяет, будет ли брандмауэр активизироваться при запуске системы. `MFW_MASQ` указывает, должен ли быть акти-

вирован маскарадинг. Остальные переменные указывают интерфейсы и адреса локальной сети или соединения с Интернетом:

```
# Файл /etc/default/myfirewall
# запуск брандмауэра: да/нет
MFW_ACTIVE=yes
# Запуск маскарадинга: да/нет
MFW_MASQ=yes
# Локальная сеть
MFW_LAN=eth1
MFW_LAN_IP=192.168.0.0/24
# Интернет
MFW_INET=eth0
```

Остановка работы брандмауэра (myfirewall-stop)

Сценарий `myfirewall-stop` восстанавливает исходное состояние `iptables` и деактивирует брандмауэр. Сначала с помощью команды `.` считывается конфигурационный файл `myfirewall`. Кроме того, `which` узнает, где находятся команды `iptables` и `sysctl`. (Каталог, в котором установлены эти команды, может отличаться в зависимости от дистрибутива.)

После этого `iptables -P` задает стандартное поведение всех фильтров в значение `ACCEPT`. Команда `iptables -F` удаляет все имеющиеся правила, причем для NAT-таблицы нужна собственная команда. `iptables -X` удаляет все определенные пользователем цепочки правил. Теперь Netfilter допускает любую передачу информации по IP.

```
#!/bin/bash
# Файл /etc/myfirewall/myfirewall-stop

# Считывание конфигурационных настроек
. /etc/default/myfirewall
IPT=$(which iptables)
SYS=$(which sysctl)

# сброс iptables
$IPT -P INPUT ACCEPT
$IPT -P OUTPUT ACCEPT
$IPT -P FORWARD ACCEPT
$IPT -P POSTROUTING ACCEPT -t nat
$IPT -P PREROUTING ACCEPT -t nat
$IPT -P OUTPUT ACCEPT -t nat
$IPT -F
$IPT -F -t nat
$IPT -X

# Остановка маскарадинга
$SYS -q -w net.ipv4.ip_forward=0
```

Не забудьте пометить сценарный файл как исполняемый с помощью `chmod u+x`!

Запуск брандмауэра (myfirewall-start)

Разумеется, сценарий `myfirewall-start` гораздо интереснее. Он начинает работу с выполнения останавливающих правил. Таким образом, происходит сброс системы Netfilter. В результате вы гарантируете для других команд `iptables`, что перед началом их работы не были определены никакие правила.

Первая команда открывает доступ к SSH-серверу из Интернета. Это правило целесообразно, если компьютер соединен с Интернетом напрямую (а не через ADSL-роутер и не через другой компьютер) и если администрирование этого компьютера должно осуществляться извне. Если кроме SSH на компьютере предоставляются и другие общедоступные службы, использование этих служб должно быть разрешено именно на данном этапе. В противном случае сформулированное ниже `wall`-правило запретит установление любых соединений извне. Если, наоборот, вы хотите заблокировать и SSH из Интернета, удалите первую команду `iptables`.

Закрытие портов

Цикл `for` полностью блокирует некоторые порты от всей информации, приходящей из Интернета. При необходимости вы можете сами дополнить этот список портов.

```
#!/bin/bash
# Файл /etc/myfirewall/myfirewall-start (часть 1)

# Считывание конфигурационных настроек
. /etc/default/myfirewall
IPT=$(which iptables)
SYS=$(which sysctl)

if [ $MFW_ACTIVE != "yes" ]; then
  echo "Firewall disabled in /etc/default/myfirewall"
  exit 0
fi

# Сброс всех правил брандмауэра
. /etc/myfirewall/myfirewall-stop

# Разрешить доступ из Интернета к SSH-серверу (порт 22)
$IPT -A INPUT -i $MFW_INET -p tcp --dport 22 -j ACCEPT

# Полностью закрыть определенные порты
#   23 (telnet)
#   69 (tftp)
#  135 (Microsoft DCOM RPC)
#  139 (NetBIOS/Samba/т.д.)
#  445 (Файловая система CIFS для Samba/SMB)
#  631 (ipp/CUPS)
# 1433 (Microsoft SQL Server)
# 2049 (NFS)
# 3306 (MySQL)
# 5999-6003 (X-Displays)
```

```
for PORT in 23 69 135 139 445 631 1433 2049 3306 \
            5999 6000 6001 6002 6003; do
  $IPT -A INPUT  -i $MFW_INET -p tcp --dport $PORT -j DROP
  $IPT -A OUTPUT -o $MFW_INET -p tcp --dport $PORT -j DROP
  $IPT -A INPUT  -i $MFW_INET -p udp --dport $PORT -j DROP
  $IPT -A OUTPUT -o $MFW_INET -p udp --dport $PORT -j DROP
done
```

Цепочка правил wall

В следующем примере задается новая цепочка правил под названием `wall`. Эта цепочка — одновременно очень изящная и эффективная защита от новых соединений, устанавливаемых извне. Первое правило `wall` гласит, что должны приниматься все пакеты, которые относятся к уже существующему соединению.

Второе правило позволяет принимать пакеты, инициирующие новое соединение, если это соединение устанавливается не через интернет-интерфейс. Инверсия синтаксически выражается с помощью восклицательного знака, стоящего перед параметром `-i`. Это правило означает, что, например, мы можем начать из локальной сети обмен информацией с компьютером по HTTP, а из Интернета — не можем.

Третье правило указывает, что все пакеты, не соответствующие предыдущим правилам, должны отклоняться. Иначе говоря, это правило можно сформулировать как «Все, что не разрешено, — запрещено!». Тогда потенциальному агрессору, которых хочет прорваться на ваш компьютер из Интернета, не удастся даже запустить SSH-соединение (разумеется, то же касается и любых других сетевых служб — HTTP, FTP, Telnet и т. д.).

Две заключительные команды сценария указывают, что все пакеты, проходящие через фильтры входа и переадресации, проверяются на основании правил `wall`:

```
# Файл /etc/myfirewall/myfirewall-start.conf (продолжение)

# Эта цепочка правил блокирует все попытки соединений,
# инициируемые извне (из Интернета)
$IPT -N wall
$IPT -A wall -m state --state ESTABLISHED,RELATED -j ACCEPT
$IPT -A wall -m state --state NEW -i ! $MFW_INET -j ACCEPT
$IPT -A wall -j DROP

# Эта цепочка правил применяется для INPUT и FORWARD
$IPT -A INPUT -j wall
$IPT -A FORWARD -j wall
```

Маскарадинг

Наконец, активизируем маскарадинг:

```
# Файл /etc/myfirewall/myfirewall-start.conf (продолжение)
# Активизация маскарадинга
if [ $MFW_MASQ = 'yes' ]; then
  $IPT -A POSTROUTING -t nat -o $MFW_INET -s $MFW_LAN_IP -j MASQUERADE
  $SYS -q -w net.ipv4.ip_forward=1
fi
```

Интеграция Upstart

Чтобы автоматически активизировать брандмауэр во время запуска компьютера до запуска сетевых интерфейсов (либо чтобы деактивировать при выключении компьютера после отключения сетевых интерфейсов), создайте собственный конфигурационный файл в /etc/init:

```
# Файл /etc/init/myfirewall.conf
description "myfirewall"
start on (starting network-interface
        or starting network-manager
        or starting networking)
stop on runlevel [!023456]
pre-start exec /etc/myfirewall/myfirewall-start
post-stop exec /etc/myfirewall/myfirewall-stop
```

С помощью service можно запустить, остановить и перезапустить брандмауэр вручную:

```
root# service myfirewall start
root# service myfirewall stop
root# service myfirewall restart
```

Интеграция Init-V

Можно активизировать брандмауэр не с помощью Upstart, а с использованием сценария Init-V. В принципе такой механизм функционирует почти во всех дистрибутивах, поскольку и Upstart, и Systemd совместимы с системой Init-V. Но детали сценария нужно откорректировать в соответствии с конкретным дистрибутивом.

```
#!/bin/sh -e
# Собственный сценарий Init-V /etc/init.d/myfirewall
### BEGIN INIT INFO
# Обеспечивает:          Брандмауэр
# Принудительный запуск: networking
# Принудительная остановка:
# Запуск по умолчанию:   S
# Краткое описание:      Запуск брандмауэра и маскарадинга
### END INIT INFO

# Базовые функции
. /lib/lsb/init-functions

# Функции для запуска, остановки и перезапуска
case "$1" in
   start|restart)
      log_begin_msg "Starting firewall and masquerading ..."
      . /etc/myfirewall/myfirewall-start
      log_end_msg 0
      ;;
   stop)
```

```
        log_begin_msg "Stopping firewall and masquerading ..."
        . /etc/myfirewall/myfirewall-stop
        log_end_msg 0
        ;;
    status)
        /sbin/iptables -L -v -n
        ;;
    *)
        log_success_msg "Usage: xxx {start|stop|restart|status}"
        exit 1
        ;;
esac
exit 0
```

Чтобы этот сценарий автоматически выполнялся при запуске системы, создайте следующую ссылку:

```
root# cd /etc/rcS.d
root# ln -s ../init.d/myfirewall S41myfirewall
```

27 Виртуальные частные сети

В наши дни беспроводной доступ к Интернету (по сетям WLAN) стал практически вездесущим, в том числе на предприятиях, на файловых серверах которых хранятся конфиденциальные данные. В настоящее время современная технология шифрования беспроводного доступа WLAN, называемая WPA2, считается надежной, но иногда она оказывается неприменимой. В частности, это происходит при работе со старыми ноутбуками и смартфонами, несовместимыми с WPA2. Привлекательным вариантом решения такой проблемы является использование *виртуальной частной сети* (VPN). При этом сама беспроводная сеть выступает своего рода носителем, на котором выстраивается вторая (виртуальная) сеть.

Сам обмен информацией по VPN является зашифрованным. Эту информацию невозможно перехватить, даже если злоумышленнику удастся взломать шифрование WLAN. В этой главе сначала изложены некоторые основы работы с VPN, а потом показано, как дополнить ваш LAN-роутер VPN-каналом для WLAN-соединений. VPN реализуется на базе технологии PPTP.

27.1. Основы VPN

Виртуальной частной сетью называется сетевое соединение двух или нескольких компьютеров поверх имеющейся сети. Этот метод наиболее примечателен тем, что сеть VPN является защищенной, тогда как лежащая в ее основе передающая среда, например Интернет или WLAN, остается незащищенной (в данном случае под словом «незащищенная» подразумевается, что злоумышленник может сравнительно просто считать передаваемую информацию и даже манипулировать ею).

«Виртуальный» в данном контексте означает, что частная сеть строится на базе уже имеющейся сети. Этот процесс часто называют термином «туннель»: в имеющейся сети создается защищенный снаружи туннель, через который происходит обмен информацией между компьютерами, образующими частную сеть. Трафик в туннеле отличается от остального сетевого трафика тем, что он использует особый протокол, особое шифрование и т. д. Далее перечислены различные методы создания виртуальных частных сетей.

«Частный» здесь означает, что новая сеть лучше защищена от окружающей среды, а значит, и от возможных вторжений. Иначе говоря, сеть является «частной»,

в отличие от условно общедоступного Интернета и недостаточно защищенных сетей WLAN.

Пользователь VPN может видеть обе сети (после того как выполнит `ifconfig`). В примерах, которые приводятся в этой книге, используется базовая (общедоступная) сеть с адресным пространством 192.168.0.*. При конфигурации клиентов очень важно то, что незащищенная базовая сеть применяется только как среда для передачи информации защищенной виртуальной частной сети.

Разумеется, шифрование данных, передаваемых по VPN, требует определенной вычислительной работы и соответствующего администрирования. Один VPN-клиент оказывает на сеть незначительную нагрузку, но общая нагрузка на процессор VPN-сервера может быть очень серьезной, если в виртуальной частной сети одновременно устанавливается много соединений.

Технологии VPN

Существует несколько возможных способов реализации VPN. В следующем списке перечислены важнейшие из них, а также упомянуты некоторые тематические сайты.

- **IPsec, Openswan.** IPsec — это протокол для безопасной передачи данных между двумя компьютерами, который может использоваться для создания виртуальной частной сети. Данный протокол — составная часть нового интернет-протокола IPv6, но IPsec подходит и для работы с действующим в настоящее время IPv4. Недостатком IPsec является его сложность: для решения даже сравнительно простых задач требуется долгая и утомительная конфигурация, поэтому мы не будем рассматривать этот протокол.

  IPsec в версии 2.6 и выше интегрирован прямо в ядро. Для управления функциями IPsec требуется в числе прочих команда `ipsec`, входящая в состав пакета `openswan`. IPsec также поддерживается многими другими операционными системами (в том числе всеми современными версиями Windows). Информацию о IPsec и Openswan вы можете найти на следующих сайтах:

  - http://www.openswan.org/;
  - http://www.ipsec-howto.org/;
  - http://lartc.org/howto/lartc.ipsec.html.

- **L2TP.** Протокол туннелирования PPP-соединения уровня 2 — плод совместной работы Microsoft и Cisco. Он объединяет принципы работы PPTP и протокола Cisco L2F (протокол эстафетной передачи на втором уровне). В самом протоколе L2TP не предусмотрено никаких надежных методов аутентификации, поэтому он применяется в комбинации с IPsec (L2TP/IPsec). При этом IPsec отвечает за аутентификацию и шифрование, а L2TP — за управление туннелем. Более подробные сведения о L2TP вы найдете на следующих сайтах:

  - http://www.jacco2.dds.nl/networking/freeswan-l2tp.html;
  - http://sourceforge.net/projects/openl2tp/.

- **PPTP.** Туннельный протокол типа «точка — точка» совмещает черты протокола «точка — точка» (PPP), который первоначально разрабатывался для

интернет-соединений через модем и ISDN, а также зашифрованного туннеля (http://www.poptop.org/). PPTP не только подходит для передачи IP-пакетов, но и совместим с другими протоколами (например, Novell IPX, AppleTalk и т. д.).

Протокол PPTP был разработан компанией Microsoft, документация по нему находится в открытом доступе (RFC 2637), поэтому интегрировать компьютеры с Windows в PPTP-VPN можно без проблем. PPTP работает и со смартфонами.

Первые версии PPTP были весьма несовершенны с точки зрения безопасности, но сегодня такие недостатки уже исправлены. Однако все-таки современная версия PPTP с точки зрения безопасности не идеальна. В последний раз аббревиатура PPTP оказалась в заголовках газет, когда один облачный сервис пообещал премию в $200 за взлом шифрования PPTP в течение 24 часов.

- **OpenVPN.** Это достаточно простой VPN-демон, не требующий специальных модулей ядра, в отличие от IPsec (http://openvpn.net/). Обмен данными происходит с помощью зашифрованных UDP-пакетов. Для доступа к трафику VPN в Linux используются специальные устройства tun и tap. Значительным достоинством OpenVPN является его сравнительно несложная конфигурация. OpenVPN работает в Linux, Windows и в различных UNIX-производных системах (Solaris, *BSD, Mac OS X).
- **PPPD и SSH.** Туннели можно создавать и с помощью SSH. Такой туннель применяется для зашифрованного обмена информацией между двумя компьютерами. Обычно SSH-туннель имеет «ширину» всего один IP-порт. Только при комбинации SSH с PPP-демоном можно направить через SSH-туннель весь IP-трафик. Создаваемая таким образом сеть VPN концептуально похожа на решение с PPTP. Сочетание SSH и PPPD подробно рассмотрено на сайтах http://www.oreillynet.com/pub/a/wireless/2001/02/23/wep.html и http://www.tldp.org/HOWTO/VPN-HOWTO/.

Перечисленные выше VPN-технологии отличаются друг от друга в первую очередь тем, на каком уровне происходит шифрование сетевого трафика. При использовании IPsec шифрование и построение туннеля осуществляется уже на уровне IP, то есть является *низкоуровневым*. В остальных вариантах для передачи зашифрованных данных, напротив, используются стандартные TCP- и UDP-пакеты. Такой подход к решению проблемы не требует вмешательства в ядро, но может вызывать проблемы, связанные с маскарадингом и некоторыми IP-протоколами (например, FTP).

VPN-технологий много, и выбрать из них не так-то просто. В этой книге рассмотрена в основном конфигурация PPTP, так как она сравнительно проста. В частной сфере PPTP обеспечивает вполне приемлемый уровень безопасности, но при работе на корпоративном уровне для PPTP лучше подыскать какую-нибудь альтернативу.

Топологии сетей VPN

Очевидно, что если существуют различные варианты реализации VPN-соединений, то есть и большое количество возможностей соединения компьютеров по VPN. В следующем списке перечислены некоторые варианты.

- **Соединение «точка — точка» между двумя компьютерами.** В этом простейшем случае мы просто создаем безопасное соединение между двумя компьютерами поверх незащищенной сети (WLAN, Интернет). Часто можно обойтись и без конфигурации VPN, применив обычный SSH.
- **Сценарий «клиент — сервер».** В данном случае многочисленные клиенты должны обращаться за информацией к центральному серверу. Такой метод часто именуется «странствующий воин» (road warrior scenario) и предназначен для сотрудников, которые работают удаленно и часто бывают в пути, а при этом им требуется безопасное соединение с сервером фирмы через ноутбук.
- **Сценарий «сервер — сервер».** Для обмена конфиденциальными данными VPN можно устанавливать между двумя серверами, которые географически находятся друг от друга достаточно далеко. Клиенты обеих сетей обмениваются информацией только с теми серверами, с которыми они соотнесены. Такой вариант топологии обычно применяется, когда следует связать два отдаленных друг от друга офиса фирмы (например, один находится в Европе, а другой — в США) через обычное интернет-соединение с помощью VPN.

В этой книге будет описан только сценарий «клиент — сервер», причем предполагается, что соединение между клиентами и сервером производится через WLAN.

Независимо от применяемого варианта может отличаться еще кое-что, а именно то, какая часть IP-трафика будет направлена через VPN: при работе с WLAN обычно бывает целесообразно пустить весь IP-трафик через VPN. При применении сценария «сервер — сервер», напротив, лучше передавать через VPN только те данные, которые важны для обеспечения безопасности (например, для синхронизации баз данных или файловых систем). Все остальные интернет-службы сервер предоставляет своим клиентам напрямую. Поскольку от данных факторов также зависят функции брандмауэра и роутера, мы имеем практически бесконечное количество конфигурационных возможностей, которые не являются предметом обсуждения этой книги. По многим VPN-решениям уже есть отдельные книги, в которых те или иные методы рассмотрены детально.

27.2. Реализация VPN с помощью PPTP

В следующих абзацах, а также на рис. 27.1, обобщены условия, необходимые для построения VPN.

- VPN-сервер устанавливается на том же компьютере, на котором находятся интернет-шлюз и брандмауэр.
- VPN-сервер служит для защиты информации, передаваемой по локальной сети WLAN (сценарий «клиент — сервер»). VPN-соединение извне (через Интернет) не допускается.
- Точка доступа WLAN подключается к серверу через специальный сетевой интерфейс. Конфигурация DHCP-сервера такова, что на этом интерфейсе компьютерам присваиваются адреса из адресного пространства 172.168.0.*.
- Трафик, идущий через сетевой интерфейс WLAN, насколько это возможно, блокируется брандмауэром. Принимаются только пакеты, которые необходимы

для DHCP-конфигурации клиентов WLAN и для VPN. WLAN-специфичная часть брандмауэра описана в подразделе «Настройка брандмауэра для PPTP-сервера» этого раздела.

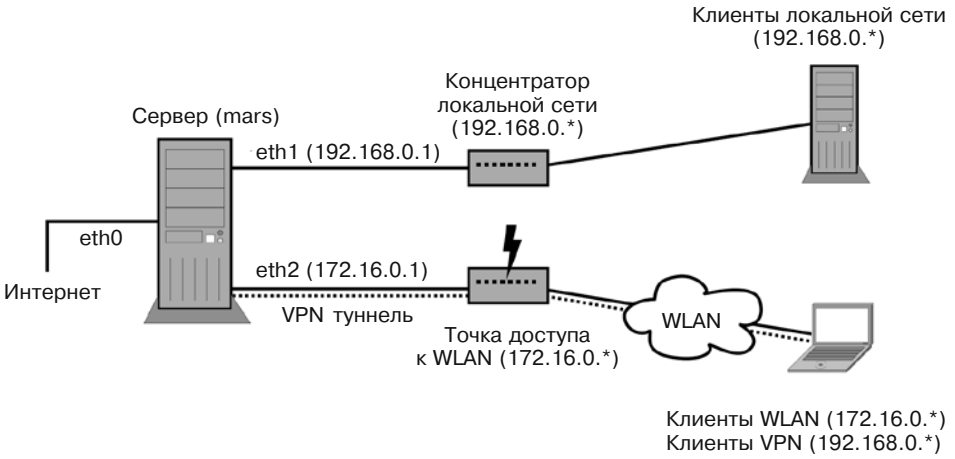


Рис. 27.1. Топология сети VPN

Локальная сеть состоит из двух частей: защищенной (LAN) и незащищенной (WLAN):

- 192.168.0.* — обычная локальная сеть (частная);
- 172.16.0.* — беспроводная сеть (общедоступная, WLAN).

На сервере нужно оборудовать *три* сетевых интерфейса, которые будут работать при подобной конфигурации следующим образом:

- `eth0` — 10.0.0.1 (доступ в Интернет через ADSL-роутер);
- `eth1` — 192.168.0.1 (LAN);
- `eth2` — 172.16.0.1 (WLAN).

Сервер использует сеть WLAN только для работы с DHCP и VPN. Все остальные сетевые функции предоставляются клиентам WLAN через VPN-туннель. Клиенты WLAN через VPN подключаются к сети 192.168.0.*. Таким образом, с помощью VPN они органично вплетаются в локальную сеть.

Проблемы, связанные с сервером имен. Принципиальное ограничение PPTP-VPN заключается в том, что хотя VPN-клиенты и могут использовать локальную сеть, их имена при этом остаются неизвестными. Причина этого проста: IP-адреса клиентам раздает не DHCP-сервер, а PPTP-сервер. А сервер имен ничего об этом не узнает. В результате сервер имен не может соотнести хост-имена VPN-клиентов и их IP-адреса.

Конфигурация сети на сервере

Теперь мы должны добавить к конфигурации сети, представленной в главе 19, третий сетевой интерфейс — `eth2`, который должен быть сконфигурирован стати-

чески и иметь IP-адрес 172.16.0.1. Если вы работаете с Debian или Ubuntu, то строки в /etc/network/interfaces должны выглядеть так:

```
# Файл /etc/network/interfaces
...
# Интерфейс для подключения к точке доступа WLAN
auto eth2
iface eth2 inet static
  address 172.16.0.1
  netmask 255.255.255.0
```

Конфигурация DHCP. Теперь DHCP-сервер должен обслуживать интерфейс WLAN eth2 и присваивать клиентам WLAN адреса из диапазона 172.16.0.*. Если вы работаете с dnsmasq, то в конфигурации, приведенной в главе 19, нужно изменить всего три строки.

Вместо interface=eth1 укажите except-interface=eth0. В результате, dnsmasq будет обслуживать *все* сетевые интерфейсы кроме eth0. Это нужно для того, чтобы dnsmasq реагировала на запросы сервера имен с интерфейсов ppp*n*.

Кроме того, во второй строке dhcp-range укажите IP-диапазон для второго интерфейса. Правильное соотнесение адресов с интерфейсами dnsmasq обычно обеспечивает самостоятельно. Наконец, нужно уменьшить адресное пространство в dhcp-range для eth1 (в данном случае — от 192.168.0.2 до 192.168.0.239). Остальными IP-адресами должен управлять PPTP-сервер.

```
# /etc/dnsmasq.conf
except-interface=eth0
dhcp-range=192.168.0.2,192.168.0.239,24h
dhcp-range=172.16.0.2,172.16.0.250,12h
...
```

Следующая команда активизирует сделанные изменения:

```
root# service dnsmasq restart
```

Настройка PPTPD-сервера

Теперь на серверном компьютере нужно установить программу PPTP-сервера, которая обычно находится в пакете pptpd. Настройки этой программы выполняются сразу в нескольких файлах.

Файл pptpd.conf

В /etc/pptpd.conf содержатся важнейшие настройки для pptpd. Обязательно нужно указать локальный IP-адрес сервера и адресное пространство VPN-клиента. В данном случае речь идет об адресах, которые нужно соединить друг с другом посредством VPN. Если вы используете в локальной сети и DHCP-сервер, убедитесь, что динамические адресные пространства серверов DHCP и PPTP не пересекаются. При необходимости откорректируйте dnsmasq.conf.

```
# /etc/pptpd.conf
option /etc/ppp/pptpd-options
```

```
localip   192.168.0.1
remoteip  192.168.0.240-253
listen    172.16.0.1
```

Параметры pptpd

В /etc/ppp/pptpd-options содержатся все PPP-специфичные параметры. Название этого файла, в принципе, может быть любым, но обязано совпадать с командой option в /etc/pptpd.conf.

```
# /etc/ppp/pptpd-options
name pptpd          # Такое же название должно быть указано
                    # во втором столбце файла chap-secrets
lock
require-mschap-v2
require-mppe-128
refuse-pap
refuse-chap
refuse-mschap
proxyarp
ms-dns 192.168.0.1  # PPTP-клиенты должны использовать этот DNS
```

Необходимо подробнее объяснить значение параметров pptpd: настройка name отвечает за верную интерпретацию логина и пароля PPP. Указанное здесь имя должно совпадать с именем, указанным во втором столбце файла chap-secrets.

Благодаря различным командам require и refuse для идентификации может использоваться только протокол mschap версии 2, разработанный Microsoft. Применяя его вместе с настройкой require-mppe-128 (128-битное шифрование), мы гарантируем максимально надежное исполнение PPTP.

Благодаря параметру proxyarp IP-адрес клиента VPN вносится в таблицу ARP (протокол разрешения адресов) сервера, поэтому всем остальным компьютерам, находящимся в сети, открывается доступ к VPN-клиенту.

С помощью ms-dns указывается адрес того сервера имен, которым VPN-клиенты должны пользоваться для разрешения сетевых имен. Хотя название этого параметра и начинается с ms (Microsoft), он действует и для клиентов, использующих Linux.

СОВЕТ

Не забывайте, что PPP-демон pppd также учитывает все параметры, содержащиеся в /etc/ppp/options (но параметры из /etc/ppp/pptpd-options имеют приоритет). Если возникнут проблемы с конфигурацией PPTP-сервера, в первую очередь посмотрите файл /etc/ppp/options. В нашем примере предполагается, что файл /etc/ppp/options пуст.

Если появятся проблемы с VPN-соединением, добавьте в pptpd-options еще одну строку с ключевым словом debug. Тогда pptpd и pppd будут записывать в /var/log/messages подробные статусные сообщения и сообщения об ошибках.

Файл chap-secrets

Последний конфигурационный файл называется /etc/ppp/chap-secrets. В нем содержатся имена и пароли, по которым PPTP-сервер опознает клиентов VPN. В этом

файле четыре столбца. В первом находится логин, в третьем — пароль. Чтобы пароль был достаточно надежным, его длина должна быть не менее 12 символов.

Во втором столбце располагается опознавательный код для сервера (см. настройку `name` в `pptpd-options`). В четвертом столбце можно указать либо звездочку, либо IP-адрес. В первом случае PPTP-сервер присвоит клиенту IP-адрес из диапазона `remoteip`, во втором — применит указанный IP-адрес, который должен находиться вне диапазона `remoteip`, во избежание конфликтов.

```
# /etc/ppp/chap-secrets
# Логин      Имя pptpd    Пароль           IP-адрес клиента
"vpnclient"  "pptpd"      "vpntestpassw"   *
```

Измененная конфигурация вступает в силу после перезапуска PPTP-сервера:

```
root# /etc/init.d/pptpd restart
```

Чтобы демон `pptpd` автоматически запускался и завершал работу при начале и окончании работы компьютера, нужно в зависимости от дистрибутива выполнить следующие команды:

```
root# chkconfig --level 35 pptpd on   (Red Hat)
root# insserv pptpd                   (SUSE)
root# systemctl enable pptpd.service  (Systemd)
```

Статус

Теперь клиентский компьютер должен установить соединение с PPTP-сервером (в разделе 18.3 описано, как это делается). При каждом удачном соединении на сервере создается новый ppp-интерфейс, то есть `ppp0`, `ppp1` и т. д.

Состояние этих интерфейсов можно узнать с помощью команды `ifconfig`. В следующих строках кратко обобщены интерфейсы, работающие на компьютере mars.sol. В данном примере `lo` — это петлевой интерфейс, `eth0` — интерфейс для соединения с Интернетом, `eth1` — интерфейс для LAN, `eth2` — интерфейс для WLAN, а `ppp0` — VPN-интерфейс для первого клиента PPTP.

```
root# ifconfig
lo   Protocol:Local loop
     inet Addr:127.0.0.1 Mask:255.0.0.0
     ...
eth0 Link encap:Ethernet-Hardware-Address 00:16:17:cd:c3:81
     inet Adr:10.0.0.1 Bcast:10.0.0.255 Mask:255.255.255.0
     ...
eth1 Link encap:Ethernet-Hardware-Address 00:14:6c:8e:d9:71
     inet Adr:192.168.0.1 Bcast:192.168.0.255 Mask:255.255.255.0
     ...
eth2 Link encap:Ethernet-Hardware-Address 00:4f:4e:0f:8e:a0
     inet Adr:172.16.0.1 Bcast:172.16.255.255 Mask:255.255.0.0
     ...
ppp0 Link encap:Point-to-Point-Connection
     inet Adr:192.168.0.1 P-z-P:192.168.0.240 Mask:255.255.255.255
     ...
```

Настройка брандмауэра для PPTP-сервера

С точки зрения безопасности рекомендуется блокировать в WLAN-интерфейсе необязательные сетевые взаимодействия. Следующая инструкция исключает DHCP- и PPTP-пакеты. DHCP использует протокол UDP через порты 67 и 68. PPTP применяет протокол UDP через порт 1723, а также собственный протокол GRE.

Необходимые правила для брандмауэра вы можете составить, используя сценарий, представленный в разделе 26.5. При этом сначала следует расширить `/etc/default/myfirewall` и добавить в переменную `MFW_WLAN` имя WLAN-интерфейса:

```
# Файл /etc/default/myfirewall
...
MFW_WLAN=eth2
```

После этого нужно добавить в файл `/etc/myfirewall/myfirewall-start` следующий код:

```
# Файл /etc/myfirewall/myfirewall-start
...
# Порты 67 и 68 для UDP-принятия (DHCP)
$IPT -A INPUT  -i $MFW_WLAN -p udp --dport 67 -j ACCEPT
$IPT -A OUTPUT -o $MFW_WLAN -p udp --dport 67 -j ACCEPT
$IPT -A INPUT  -i $MFW_WLAN -p udp --dport 68 -j ACCEPT
$IPT -A OUTPUT -o $MFW_WLAN -p udp --dport 68 -j ACCEPT

# Порт 1723 для TCP-принятия (PPTP-контроль)
# Протокол GRE-принятия (выяснение 47, PPTP-данные)
$IPT -A INPUT  -i $MFW_WLAN -p tcp --dport 1723 -j ACCEPT
$IPT -A OUTPUT -o $MFW_WLAN -p tcp --sport 1723 -j ACCEPT
$IPT -A INPUT  -i $MFW_WLAN -p gre -j ACCEPT
$IPT -A OUTPUT -o $MFW_WLAN -p gre -j ACCEPT

# Блокировать все другие порты в WLAN-интерфейсе
$IPT -A INPUT  -i $MFW_WLAN -p tcp -j DROP
$IPT -A OUTPUT -o $MFW_WLAN -p tcp -j DROP
$IPT -A INPUT  -i $MFW_WLAN -p udp -j DROP
$IPT -A OUTPUT -o $MFW_WLAN -p udp -j DROP
```

27.3. Конфигурация VPN-клиента (PPTP)

Предпосылки

При конфигурации VPN-сервера, описанной в предыдущем разделе, на клиентском компьютере должны выполняться следующие условия (см. также рис. 27.1):

- компьютер, с которым вы устанавливаете соединение, должен быть сконфигурирован как PPTP-сервер;
- ваш компьютер должен быть подключен к незащищенной базовой сети, например через LAN или WLAN. Эта часть конфигурации автоматически осущест-

вляется посредством DHCP. Незащищенная базовая сеть использует адресное пространство 172.16.0.*;

- защищенная сеть (VPN) применяет адресное пространство 192.168.0.*;
- VPN-сервер доступен по адресу 172.16.0.1 (WLAN) или 192.168.0.1 (VPN).

Конфигурация в сетевом менеджере

Проще всего осуществить конфигурацию с помощью сетевого менеджера. Для этой программы существуют дополнительные модули, предназначенные для определенных методов работы с VPN, в частности, с PPTP. Но в некоторых дистрибутивах нужные пакеты приходится устанавливать дополнительно. В Ubuntu эти расширения находятся в пакетах `network-manager-pptp` и `network-manager-gnome-pptp`.

В сетевом менеджере необходимо выполнить команды Соединения ▸ Сконфигурировать. С помощью Добавить начните определение нового VPN-соединения. В списке типов VPN-соединений выберите тип Протокол туннелирования точка-точка. На вкладке VPN в качестве шлюза укажите IP-адрес PPTP-сервера (в данном случае — 172.16.0.1). В поле имени пользователя задайте `vpnclient`, в поле пароля — соответствующую последовательность символов из файла `/etc/ppp/chap-secrets` PPTP-сервера. С помощью кнопки Дополнительно откройте дополнительное конфигурационное диалоговое окно и установите там флажок Использовать шифрование MPPE.

Чтобы настроить VPN-соединение, нужно сначала установить в сетевом менеджере соединение с WLAN. В дальнейшем этот этап будет происходить автоматически, как только ваш компьютер окажется в зоне действия сети WLAN. Затем выполните VPN-соединения ▸ XXX, где XXX — имя вашего VPN-соединения.

Ручная конфигурация

Для установки клиентского PPTP-соединения применяются две тесно взаимосвязанные программы: PPP-демон `pppd` и PPTP-клиент `pptp`, который в некоторых дистрибутивах требуется предварительно установить. Пакет называется `pptp-linux`.

При конфигурации вручную нужно изменить либо заново создать два файла: `/etc/ppp/peers/vpn` и `/etc/ppp/chap-secrets`.

Ppp/peers/vpn. в файле `/etc/ppp/peers/vpn` содержатся PPP-параметры для установления VPN-соединения. Имя файла значения не имеет, но он должен находиться в каталоге `/etc/ppp/peers/`. Если VPN-сервер сконфигурирован так, как описано в разделе 27.2, то клиент должен содержать следующие параметры:

```
# /etc/ppp/peers/vpn
pty "/usr/sbin/pptp 172.16.0.1 --nolaunchpppd"
user "vpnclient"
noauth
require-mppe-128
defaultroute
usepeerdns
ipparam "vpn"   # только для Red Hat и Fedora
```

Благодаря настройке pty PPP-демон при установлении соединения сразу воспользуется программой pptp. Она обеспечивает соединение между VPN-сервером и IP-адресом 172.16.0.1.

Для аутентификации применяется логин vpnclient. Пароль для этого логина находится в файле /etc/ppp/chap-secrets (см. ниже). Та же информация о логине и пароле должна располагаться в файле chap-secrets на VPN-сервере.

Запись noauth означает, что на противоположном от VPN-сервера конце PPP-соединения аутентификации не требуется. Но сам-то себя клиент сможет идентифицировать!

Благодаря require-mppe-128 применяется разновидность шифрования Microsoft «точка — точка» со 128-битным ключом.

При использовании defaultroute IP-адрес, присвоенный VPN-серверу, применяется в качестве стандартной цели маршрутизации для IP-пакетов. Таким образом, все IP-пакеты автоматически передаются именно через сетевой интерфейс VPN, а не через какой-то другой. Но учтите, что pppd не заменяет уже имеющегося стандартного направления маршрутизации! Поэтому при необходимости перед установлением VPN-соединения может потребоваться выполнить route del default.

Благодаря usepeerdns адрес сервера имен, присвоенный VPN-сервером, записывается в файл /etc/resolv.conf. Однако в Red Hat и Fedora этот механизм работает лишь в том случае, когда с ipparam передается дополнительный параметр vpn. Соответственно, должен существовать файл /etc/sysconfig/network-scripts/ifcfg-vpn, где должна содержаться команда PEERDNS=yes.

ПРИМЕЧАНИЕ

Обратите внимание на то, что PPP-демон pppd учитывает при работе все параметры в файле /etc/ppp/options, а в некоторых дистрибутивах — и в /etc/ppp/options.pptp. Однако параметры из файла /etc/ppp/peers/vpn имеют приоритет. В этом разделе я исхожу из того, что файлы options пусты.

Если возникнут проблемы с конфигурацией PPTP-доступа, первым делом посмотрите в /etc/ppp/options и попробуйте удалить там все параметры. В SUSE также необходимо удалить команду idle 600. Под влиянием этого параметра любое PPP-соединение завершается в случае десятиминутного отсутствия активности. При модемном соединении это, пожалуй, целесообразно, но в случае с VPN — нет. Описание важнейших параметров PPP приводится в разделе 18.7.

Ppp-chap-secrets. В файле /etc/ppp/chap-secrets укажите логин и пароль для VPN-соединения (аналогичный файл с сервера PPTP приведен в пункте «Файл chap-secrets» подраздела «Настройка PPTPD-сервера» раздела 27.2).

```
# /etc/ppp/chap-secrets
#login name     server    password        IP adress
"vpnclient"     *         "vpntestpassw"  *
```

Установка и разрыв VPN-соединения. Чтобы установить соединение, выполните две приведенные ниже команды. Команда route требуется лишь в том случае, если уже существует стандартное направление маршрутизации (то есть, если route -n выдает запись, в которой в столбце flags указано UG).

```
root# route del default
root# pppd call vpn
```

Если в окне терминала начинают появляться разные непонятные сообщения, это может означать, что в /etc/ppp/options содержится параметр nodetach. Удалите этот параметр, чтобы pppd выполнялся в качестве фонового процесса.

Когда все получится, команды ifconfig и route должны выдавать следующие результаты:

```
root# ifconfig
   lo Link encap:Local Loop
   inet Address:127.0.0.1 Mask:255.0.0.0
   ...

eth1 Link encap:Ethernet Hardware Address 00:0C:F1:58:F9:93
   inet Address:172.16.0.199 Bcast:172.16.0.255 Mask:255.255.255.0
   ...

ppp0 Link encap: Point-to-Point connection
   inet Address:192.168.0.200 P-z-P:192.168.0.1 Mask:255.255.255.255
   ...

root# route -n
Kernel IP Routing table
Target          Router       Genmask         Flags  Metric    Ref    Use    Iface
192.168.0.1     0.0.0.0      255.255.255.255 UH     0         0      0      ppp0
172.16.0.0      0.0.0.0      255.255   255.0 U      0         0      0      eth1
0.0.0.0         192.168.0.1  0.0.0.0         UG     0         0      0      ppp0
```

Это означает, что, кроме обратного петлевого интерфейса, работают еще два сетевых интерфейса: eth1 с соединением в незащищенной сети WLAN и ppp0 с соединением в сети VPN. Таким образом, локальный компьютер доступен сразу по трем IP-адресам: 127.0.0.1 (обратный петлевой интерфейс), 172.16.0.199 (WLAN) и 192.168.0.200 (локальная сеть).

Весь сетевой трафик направляется через интерфейс ppp0, то есть через VPN. Исключение составляют лишь те пакеты, которые направлены непосредственно на адрес 172.16.0.* — они идут на интерфейс eth1.

Если вы хотите вновь разорвать VPN-соединение, выполните такую команду:

```
root# killall pppd
```

СОВЕТ

Если возникнут проблемы при установке VPN-соединения, нужно дополнить /etc/ppp/peers/vpn еще одной строкой с ключевым словом debug. Тогда pppd будет выводить в файле лога подробные статусные сообщения и сообщения об ошибках (как правило, речь идет о файле /var/log/messages или /var/log/syslog). Подробное руководство по поиску ошибок дано на сайте http://pptpclient.sourceforge.net/howto-diagnosis.phtml.

Запуск VPN без привилегий администратора. К сожалению, VPN-соединение может устанавливать только администратор. В Debian и Ubuntu обычные пользователи могут обходить эту проблему, прибегая к командам pon pptp или poff pptp. В остальных дистрибутивах при необходимости виртуального частного соединения нужно написать небольшой сценарий. После этого сконфигурируйте /etc/sudoers так, чтоб обычные пользователи могли выполнять такие сценарии с помощью sudo.

Конфигурация брандмауэра для PPTP-клиента

Чтобы исключить возможность того, что сетевые службы, работающие на клиентских компьютерах (например, Samba или сервер MySQL), передадут через WLAN важные данные, можно защитить интерфейс WLAN брандмауэром. Правила очень похожи на правила для сервера, представленные выше. Отличия есть только в том, как осуществляется обмен данными. Соответствующий сценарий может выглядеть так:

```
#!/bin/sh
IPT=/sbin/iptables
WLAN=eth1
# Принять порты 67 и 68 для UDP (DHCP)
$IPT   -A INPUT    -i   $WLAN   -p   udp   --dport   67   -j ACCEPT
$IPT   -A OUTPUT   -o   $WLAN   -p   udp   --dport   67   -j ACCEPT
$IPT   -A INPUT    -i   $WLAN   -p   udp   --dport   68   -j ACCEPT
$IPT   -A OUTPUT   -o   $WLAN   -p   udp   --dport   68   -j ACCEPT
# Принять порт 1723 для TCP (PPTP control)
# Принять протокол GRE
$IPT   -A INPUT    -i   $WLAN   -p   tcp   --sport   1723   -j ACCEPT
$IPT   -A OUTPUT   -o   $WLAN   -p   tcp   --dport   1723   -j ACCEPT
$IPT   -A INPUT    -i   $WLAN   -p   gre   -j ACCEPT
$IPT   -A OUTPUT   -o   $WLAN   -p   gre   -j ACCEPT
# Блокировать все остальное
$IPT   -A INPUT    -i   $WLAN   -p   tcp   -j DROP
$IPT   -A OUTPUT   -o   $WLAN   -p   tcp   -j DROP
$IPT   -A INPUT    -i   $WLAN   -p   udp   -j DROP
$IPT   -A OUTPUT   -o   $WLAN   -p   udp   -j DROP
```

28 Squid и DansGuardian (сетевой фильтр)

Squid — это так называемый кэш-посредник. Это означает, что данная программа сохраняет страницы в особый буфер обмена, расположенный на локальном компьютере, и управляет доступом к ним. Squid может выполнять три функции.

- **Контроль доступа.** Squid может полностью заблокировать определенные страницы для конкретных пользователей, предоставлять доступ к Интернету лишь в определенные часы и т. д. Для распознавания «опасных» сайтов Squid обычно используют вместе с сетевым фильтром DansGuardian. Таким образом, Squid и DansGuardian помогают ограничить доступ в Интернет дома или в общественных местах (например, в школах) и сделать его более безопасным.
- **Кэш.** Если несколько пользователей обращаются к одной и той же веб-странице через кэш-посредник, то такая программа позволяет избежать многократной передачи одного и того же файла. Это ускоряет процесс чтения часто используемых сайтов и уменьшает трафик, идущий к провайдеру. Сегодня, в эпоху Web 2.0, вы все равно не ощутите ни значительного ускорения передачи данных, ни снижения объема закачек.
- **Журналирование/наблюдение.** Squid обеспечивает централизованное наблюдение за локальной сетью — в частности, позволяет узнать, кто какие сайты посещает и насколько часто. В некоторых фирмах, где требования к безопасности очень высоки, это может быть оправданно.

В этой главе мы уделим основное внимание первому из трех перечисленных пунктов и лишь бегло коснемся функций кэширования и журналирования. Чтобы избавить вас от возможных разочарований, сразу предупреждаю: Squid и DansGuardian чудес не делают и никак не отменяют воспитательной и просветительской работы с пользователями!

На сайте Squid выложено пособие по настройке программы, а также очень подробный документ со списком часто задаваемых вопросов: http://www.squid-cache.org.

28.1. Squid

Минимальная конфигурация. Работа Squid управляется с помощью файла `/etc/squid/squid.conf`. Этот конфигурационный файл, поставляемый вместе со Squid, отлично документирован, но ужасающе велик — содержит почти 6000 строк!

Чтобы вам было легче в нем ориентироваться, сохраните резервную копию оригинального конфигурационного файла, а потом удалите все комментарии (grep с помощью параметра -v может отфильтровать все строки, которые начинаются с символа комментария #, а команда cat удалит все пустые строки). Тогда конфигурационный файл ужмется всего до 50 строк.

```
root# cd /etc/squid3
root# mv squid.conf squid.conf.orig
root# grep -v '^#' squid.conf.orig | cat -s > squid.conf
```

По умолчанию squid.conf настроен так, что кэшем может пользоваться только компьютер localhost, причем для буфера обмена резервируется 8 Мбайт оперативной памяти и 100 Мбайт на жестком диске, а каждое обращение к /var/spool/squid заносится в файл регистрации.

Если вы собираетесь использовать Squid в первую очередь как сетевой фильтр, то можно отключить функции кэширования и журналирования. Кроме того, вам следует открыть всем компьютерам локальной сети доступ к Squid. Для этого с помощью acl localnet задайте переменную localnet, а в ней укажите адресное пространство локальной сети. (В некоторых дистрибутивах установочный сценарий Squid пытается самостоятельно обнаружить локальные сети и сразу же добавляет эти команды.)

Комбинация с http_access позволяет всем компьютерам локальной сети использовать кэш (вместо localnet можно задавать любое другое имя). В следующем фрагменте изменения, внесенные в конфигурационный файл, поставляемый с Ubuntu, выделены полужирным шрифтом:

```
# /etc/squid/squid.conf
# Пример настройки для использования программы
# в качестве сетевого фильтра

# Хост-имя компьютера, на котором работает squid
visible_hostname mars.sol

# Определения
acl manager      proto cache object
acl localhost    src   127.0.0.1/32 ::1
acl to_localhost dst   127.0.0.0/8 0.0.0.0/32 ::1
acl localnet     src   192.168.0.0/255.255.255.0
acl localnet     src   192.168.0.0/255.255.255.0

acl SSL_ports port 443
acl Safe_ports port 80          # http
acl Safe_ports port 21          # ftp
acl Safe_ports port 443         # https
acl Safe_ports port 70          # gopher

acl Safe_ports port 210         # wais
acl Safe_ports port 1025-65535  # Незарегистрированные порты
acl Safe_ports port 280         # http-mgmt
acl Safe_ports port 488         # gss-http
```

```
acl Safe_ports port 591         # filemaker
acl Safe_ports port 777         # multiling http
acl CONNECT method CONNECT

# Правила доступа
http_access allow manager localhost
http_access deny manager
http_access deny !Safe_ports
http_access deny CONNECT !SSL_ports
http_access allow localhost
http_access allow localnet
http_access deny all

# Логирование отсутствует
access_log none

# Стандартный порт
http_port 3128

# Сохранение информации об аварийных завершениях
coredump_dir /var/spool/squid3

# Длительность существования файлов без даты окончания действия
refresh_pattern ^ftp:                    1440   20%    10080
refresh_pattern ^gopher:                 1440   0%     1440
refresh_pattern (Release|Packages(.gz)*)$   0      20%    2880
refresh_pattern .                        0      20%    4320

# Ограничение размера кэша
cache_mem 256 MB

# Отсутствие кэша жесткого диска:
# раскомментируем и удаляем все команды cache_dir
```

Для того чтобы внесенные изменения вступили в силу, необходимо перезапустить Squid:

```
root# service squid3 restart
```

Если ваш провайдер предлагает кэш-посредник в качестве отдельной услуги и вы можете либо обязаны его применять, то укажите его IP-адрес и порт с помощью ключевого слова `cache_peer` (как в следующем примере). Если номер ICP-порта неизвестен, попробуйте 0 или 7. Если не получается, дополнительно задайте параметр `default` или `no-query`.

```
# Вышестоящий кэш-посредник (например, интернет-провайдера)
# cache_peer <хост-имя> <тип> <прокси-порт> <icp-порт> <параметры>
cache_peer www-proxy.provider.de parent 8080 7 default
```

Если вы хотите, чтобы сайты локального сервера не сохранялись в буфере обмена, используйте ключевое слово `no_cache`. В следующем примере предполагается, что на компьютере uranus работает локальный веб-сервер:

```
# Прямой доступ к локальному веб-серверу uranus.sol
acl       mars dstdomain .uranus .uranus.sol
no_cache deny uranus
```

ПРИМЕЧАНИЕ

Есть еще не меньше сотни параметров, которые могут влиять на работу Squid или на его функции журналирования. Если вы имеете на этот счет особые пожелания, посмотрите файл squid.conf или документацию по адресу www.squid-cache.org.

Первое испытание. По умолчанию Squid использует порт 3128. Чтобы пользователи вашей локальной сети действительно могли работать с прокси, они должны изменить его конфигурацию в своих браузерах. В Firefox можно открыть Инструменты ▸ Настройки ▸ Дополнительные ▸ Сеть, нажать кнопку Настроить и установить флажок Ручная настройка сервиса прокси. В качестве прокси-сервера требуется указать тот компьютер, на котором работает Squid (например, mars.sol), номер порта — 3128.

Если вы не отключили в Squid функции журналирования, то в файле `/var/log/squid/access.log` можно проверить, правильно ли работает программа. Если все нормально, то все переданные данные будут зарегистрированы в этом файле.

28.2. Конфигурация прозрачного кэш-посредника

Пока использование кэш-посредника происходит на добровольной основе. Клиентам (пользователям Интернета) нужно просто настроить связь по прокси в своем браузере, чтобы Squid начал работать. Если настройки прокси не изменить, то пользователи будут работать как бы «мимо кэша».

Такая ситуация нас не устраивает по двум причинам: многие конечные пользователи просто не сумеют сами изменить настройки прокси. Кроме того, чтобы Squid перестал фильтровать трафик, пользователям понадобится сбросить настройки прокси.

Избежать таких неприятностей можно, настроив ядро Linux так, чтобы весь трафик HTTP, который обычно идет через IP-порт 80, автоматически перенаправлялся на кэш-посредник. Такой кэш называется прозрачным. Это означает, что кэш без всякой дополнительной конфигурации автоматически применяется пользователем при каждом обращении к HTTP. Для описанной здесь конфигурации необходимо, чтобы Squid был установлен на том же компьютере, на котором работает интернет-шлюз локальной сети.

Файл squid.conf. В `squid.conf` необходимо изменить всего одну строку. В `http_port` дополнительно к номеру порта укажите ключевое слово `transparent`, а потом перезапустите Squid.

```
# Изменение в /etc/squid/squid.conf
...
http_port 3128 transparent
```

Активизация переадресации. Если вы еще этого не сделали, когда настраивали конфигурацию маскарадинга, можете активизировать функцию переадресации в ядре:

```
root# echo 1 > /proc/sys/net/ipv4/ip_forward
```

Переадресация портов. Теперь нужно активизировать переадресацию IP-пакетов: все IP-пакеты, которые должны уйти с компьютера и адресованы на порт 80, следует переадресовать на IP-порт, где работает Squid. Команда не помещается на одной строке, поэтому я разделил ее на две части. Интерфейс `eth1` заменяется интерфейсом, через который шлюз соединяется с локальной сетью.

```
root# iptables -A PREROUTING -t nat -i eth1 -p tcp \
                --dport 80 -j REDIRECT --to-port 3128
```

Если все сделано правильно, то теперь каждый клиент, работающий в сети, автоматически использует Squid для передачи веб-страниц. Проверьте, на самом ли деле при доступе клиентского компьютера к Интернету в `/var/log/squid/access.log` появляются новые записи.

В заключение вставьте команду `iptables` в сценарий Init-V, который автоматически выполняется при запуске компьютера. На моем сервере эта команда интегрирована в сценарий брандмауэра (см. начало главы 26):

```
# в /etc/myfirewall/myfirewall-start.conf
...
if [ $MFW_MASQ = 'yes' ]; then
  # Маскарадинг
  $IPT -A POSTROUTING -t nat -o $MFW_INET -s $MFW_LAN_IP \
       -j MASQUERADE

  # Прозрачный кэш-посредник
  $IPT -A PREROUTING -t nat -i eth1 -p tcp --dport 80 \
       -j REDIRECT --to-port 3128

  # Функция переадресации в ядре
  $SYS -q -w net.ipv4.ip_forward=1
fi
```

ВНИМАНИЕ

Принцип работы конфигурации таков, что прокси функционирует только на клиентских компьютерах локальной сети, но не работает на том компьютере, где установлен прозрачный прокси! Другими словами: автоматическая защита прокси не действует на локальном компьютере, но применяется на всех остальных компьютерах в сети. Чтобы использовать прокси и на своем локальном компьютере, вам потребуется изменить на нем в браузере настройки прокси.

Переадресация портов для VPN. Представленный выше механизм переадресации портов, к сожалению, не работает с такими компьютерами, которые подключены к вашей локальной сети с помощью VPN (см. предыдущую главу). Причина заключается в следующем: переадресация портов действует только для такого сетевого трафика, который проходит непосредственно через интерфейс `eth1` (здесь должно быть имя того сетевого интерфейса, который соединяет ваш компьютер с локальной сетью). Сетевой трафик VPN-клиентов, напротив, проходит через `ppp`*n*-порты.

Чтобы справиться с этой проблемой, нужно дополнить сценарий брандмауэра определением универсального правила для `ppp`*n*-портов.

```
# в /etc/myfirewall/myfirewall
...
if [ $MFW_MASQ = 'yes' ]; then
   ...
   # прозрачный кэш-посредник
   $IPT -A PREROUTING -t nat -i $MFW_LAN -p tcp --dport 80 \
      -j REDIRECT --to-port 3128
   $IPT -A PREROUTING -t nat -i ppp+ -p tcp --dport 80 \
      -j REDIRECT --to-port 3128
fi
```

28.3. DansGuardian

Правила `squid.conf` позволяют блокировать отдельные веб-страницы. Но создать по такому принципу полноценный сетевой фильтр будет очень сложно. Правда, нет необходимости заново что-то изобретать. Программа DansGuardian позволяет защитить ваших детей от опасности, которой в наши дни в Интернете предостаточно.

Основная функция этой программы заключается в том, что она анализирует текст сайтов на предмет наличия в них слов, связанных с порнографией, и блокирует переадресацию сайта после превышения определенной границы. Этот механизм достаточно интеллектуален. Поскольку DansGuardian перерабатывает содержимое сайтов, программа работает независимо от списков заблокированных сайтов, которые могут устаревать. Обзор других возможностей этой программы приводится на сайте http://dansguardian.org/.

Установка. В большинстве дистрибутивов DansGuardian можно установить в виде отдельного пакета. Поскольку программа также блокирует загрузку большинства известных вирусов, вместе с DansGuardian инсталлируется вирусный фильтр ClamAV (эта программа предназначена для защиты клиентов Windows, работающих в локальной сети; для Linux в настоящее время вирусов не существует).

DansGuardian — это демон, запускаемый и останавливаемый обычными командами. По умолчанию программа ожидает поступления веб-запросов на порт 8080. Если такой запрос поступает, то DansGuardian связывается с программой Squid (порт 3128) и анализирует сайт. Если эта страница окажется безобидной, демон переадресует ее через порт 8080 обратно в браузер. Еще более изящным является представленный ниже вариант конфигурации, при котором DansGuardian обрабатывает все запросы на порте 80 и по этой причине для клиентов не требуется никакой настройки прокси.

Файл dansguardian.conf. Основным конфигурационным файлом программы является `/etc/dansguardian/dansguardian.conf`. Как правило, его можно оставить без изменений — в том виде, в котором он был при поставке системы. В зависимости от дистрибутива в файле может содержаться строка `UNCONFIGURED`; если она есть, то перед ней нужно поставить символ комментария. Если вы хотите, чтобы сообщение `'access denied'` выводилось по-русски, используйте настройку `language='russian'`. Если в вашей сети все компьютеры работают с Linux, отключите с помощью `virusscan=off` сканер антивируса, который по умолчанию проверяет все закачки на

наличие вирусов. Для обеспечения взаимодействия с прокси Squid нужно указать IP-адрес и номер порта.

```
# /etc/dansguardian/dansguardian.conf (выборочно)
# UNCONFIGURED (перед этой строкой должен стоять знак комментария!)
reportinglevel = 3
languagedir = '/etc/dansguardian/languages'
language = 'german'
loglevel = 1
logexceptionhits = 2
logfileformat = 1
filterport = 8080
proxyip = 127.0.0.1
proxyport = 3128
...
```

Завершив конфигурацию, запустите DansGuardian:

```
root# service dansguardian start
```

Прозрачная защита. По умолчанию DansGuardian работает только с обращениями, поступающими на порт 8080. Чтобы обеспечить прозрачную защиту всех обращений к Интернету, идущих через порт 80, нужно еще раз немного изменить сценарий брандмауэра. Вместо переадресации HTTP-обращений на порт 3128 (Squid) выполните переадресацию на порт 8080 (DansGuardian):

```
# Дополнение сценария брандмауэра
#
# Переадресовывать к DansGuardian (Port 8080)
# все пакеты, идущие на порт 80 (прозрачный сетевой фильтр)
iptables -t nat -A PREROUTING -i eth1 -p tcp --dport 80 \
         -j REDIRECT --to-port 8080
```

Как это уже было описано выше, Squid нужно сконфигурировать как прозрачный кэш-посредник. Кроме того, доступ к Squid нужно ограничить таким образом, чтобы со Squid мог обмениваться информацией только тот компьютер, на котором работает DansGuardian. Если и Squid, и DansGuardian работают на одном и том же компьютере и вы применяете конфигурацию, описанную в этой главе, нужно удалить строку `http_access allow localnet`. В противном случае все пользователи локальной сети смогут так настроить конфигурацию прокси в своем браузере, что браузер будет обмениваться данными напрямую со Squid через порт 3128 и обходить таким образом DansGuardian!

Конфигурация сетевого фильтра

Сами фильтрующие функции DansGuardian управляются из файла `/etc/dansguardian/dansguardianf1.conf`. Сначала этот файл указывает на различные заранее сконфигурированные файлы, содержащие ключевые слова, заблокированные сайты и т. д. Параметр `naughtynesslimit` определяет границу, начиная с которой сайт блокируется. Это значение тем выше, чем больше ключевых слов или их комбинаций будет найдено в тексте.

```
# /etc/dansguardian/dansguardianf1.conf (выборочно)
# Размещение файлов, предназначенных для фильтрации содержимого
bannedphraselist    = '/etc/dansguardian/bannedphraselist'
weightedphraselist  = '/etc/dansguardian/weightedphraselist'
exceptionphraselist = '/etc/dansguardian/exceptionphraselist'
bannedsitelist      = '/etc/dansguardian/bannedsitelist'
greysitelist        = '/etc/dansguardian/greysitelist'
exceptionsitelist   = '/etc/dansguardian/exceptionsitelist'
bannedurllist       = '/etc/dansguardian/bannedurllist'
greyurllist         = '/etc/dansguardian/greyurllist'
exceptionurllist    = '/etc/dansguardian/exceptionurllist'
bannedregexpurllist = '/etc/dansguardian/bannedregexpurllist'
bannedextensionlist = '/etc/dansguardian/bannedextensionlist'
bannedmimetypelist  = '/etc/dansguardian/bannedmimetypelist'
picsfile            = '/etc/dansguardian/pics'
contentregexplist   = '/etc/dansguardian/contentregexplist'
urlregexplist       = '/etc/dansguardian/lists/urlregexplist'

# Лимит пошлости
# в качестве ориентира:
# 50 для маленьких детей, 100 для школьников,
# 160 для старших тинейджеров.
naughtynesslimit = 50
...
```

Если DansGuardian блокирует сайт, то браузер выводит соответствующее сообщение (рис. 28.1). Внешний вид и содержимое страницы, открывающейся при блокировке контента, можно настроить в файле `/etc/dansguardian/languages/german/template.html`.

В табл. 28.1 перечислены важнейшие файлы, находящиеся в каталоге `/etc/dansguardian`. В этих файлах, а также в названных в них включаемых файлах можно перечислить и другие понятия, сайты и типы файлов.

Таблица 28.1. Конфигурационные файлы DansGuardian

Файл	Содержание
bannedextensionlist	Блокировка файлов с указанными расширениями
bannedmimetypelist	Блокировка файлов заданных типов
bannedphraselist	Запрещенные ключевые слова
weightedphraselist	Негативные ключевые слова
bannedsitelist	Полностью блокировать указанные сайты
bannedurllist	Блокировать эти страницы
exceptioniplist	Никогда не блокировать запросы, поступающие с этих компьютеров (административный доступ к сомнительным сайтам)
exceptionsitelist	Принимать такие сайты без дополнительного контроля
exceptionurllist	Принимать такие страницы без дополнительного контроля
exceptionphraselist	Положительные ключевые слова
greysitelist	Принимать такие сайты, но проверять их текстовое содержимое

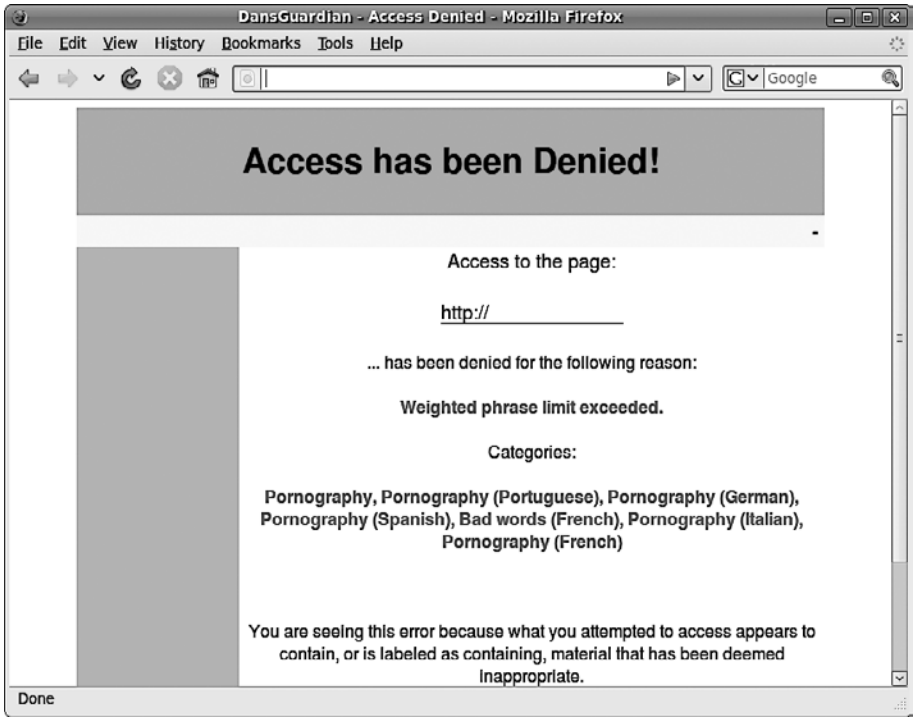


Рис. 28.1. DansGuardian и Squid блокируют доступ к сайту

При анализе текста DansGuardian различает запрещенные ключевые слова (`bannedphraselist`), при обнаружении которых доступ к странице сразу же блокируется, и подозрительные ключевые слова (`weightedphraselist` и `exceptionphraselist`). В базовой конфигурации полностью запрещенных ключевых слов нет, так как самые подозрительные слова иногда могут попадаться на совершенно безобидных сайтах. Вместо этого DansGuardian применяет довольно обширные списки слов (см. файл `/etc/dansguardian/phraselists`). На основании этих списков программа подводит суммарный итог встречаемости всех негативных и положительных ключевых слов. Если итог превышает заданное пороговое значение, то доступ блокируется. Подробности о том, как вычисляются оценочные суммы блокируемых сайтов, описаны в файле регистрации `/var/log/dansguardian/access.log`.

В пакете DansGuardian *нет* списков сетевых адресов, которые должны блокироваться! В файлах `bannedsitelist` и `bannedurllist` документированы только принципы построения синтаксиса. Но есть фирмы, которые позволяют регулярно скачивать обновляемые списки за определенную плату. Самая известная из них — URLblacklist (http://urlblacklist.com/), собирающая в различных Squid-совместимых текстовых файлах ссылки на неблагонадежные сайты. Списки относятся к различным категориям. Фильтрационные списки можно бесплатно скачивать, чтобы научиться с ними работать, но обновление этих списков платное.

При использовании стандартной конфигурации DansGuardian блокирует в числе прочих закачки заархивированных файлов, MP3, образов дисков (ISO) и т. д.

Но для повседневной работы с Linux такие ограничения определенно слишком жесткие, а в некоторых дистрибутивах они даже мешают работе системы управления пакетами. Просмотрите файлы `bannedmimetypelist` и `bannedextensionlist` и поставьте символы комментария перед всеми типами файлов, загрузку которых вы хотите разрешить. Вы также можете, настраивая переменные `bannedextensionlist` и `bannedmimetypelist`, просто указать в `dansguardianf1.conf` пустой файл.

Чтобы реагировать на запросы типа DansGuardian блокирует сайт xxx, но он совершенно безобиден, вы как администратор должны уметь обходить DansGuardian. Это проще всего сделать, указав в конфигурационном файле `exceptionip` фиксированный IP-адрес вашего компьютера в локальной сети.

Ограничения

Не переоценивайте эффективность DansGuardian! Даже настройка `naughtynesslimit = 50` (для маленьких детей) не гарантирует полной защиты от порнографических сайтов и совсем не дает защиты от порнографических изображений. В частности, можно задавать следующие ограничения.

- При стандартной конфигурации учитываются в основном английские ключевые слова. Для блокировки большинства порнографических сайтов этого обычно бывает достаточно, но нужно также заблокировать экстремистские правые сайты и сайты, призывающие к насилию, и для этого придется поработать вручную. Начать нужно с файла `/etc/dansguardian/weightedphraselist`.
- Независимо от этого, фильтр легко обходится динамически создаваемым текстом (JavaScript/Flash), а также не воспринимает содержимого внешних файлов (PDF и т. д.).
- Доступ к изображениям и видео остается полностью незащищенным.
- Squid и DansGuardian работают только с обычными сайтами. Другие виды обмена информацией (электронная почта, новости, чат) не защищены.
- Незащищенными остаются и HTTPS-сайты. Squid в принципе не предназначен для обработки HTTPS-сайтов.

29 SELinux и AppArmor

SELinux и AppArmor — это расширения ядра, отслеживающие текущие процессы и гарантирующие, что будут соблюдаться определенные правила. SELinux по умолчанию применяется в Fedora и Red Hat AppArmor — в Ubuntu и частично в дистрибутивах SUSE.

В этой главе описываются принципы функционирования и конфигурация SELinux и AppArmor. Кроме того, здесь показано, как обнаруживать нарушения правил и реагировать на них.

29.1. SELinux

Обычно в Linux используется традиционная система управления правами доступа: каждая программа выполняется под учетной записью определенного пользователя. На основании учетной записи система узнает, к каким файлам (и файлам-устройствам) может обращаться программа.

Обычные приложения работают с учетной записью пользователя, запустившего ту или иную программу. Сетевые службы, серверы баз данных и т. д. запускаются с учетной записи администратора, но из соображений безопасности практически сразу после запуска переводятся на другую учетную запись с ограниченными правами.

Система прав в UNIX очень проста, но возможности ее конфигурации ограничены. Если злоумышленнику удастся перехватить управление одной из ваших программ, то он сможет получить доступ к огромному количеству данных, которые программе обычно не требуются. Ситуация особенно осложняется, если злоумышленнику удается завладеть правами администратора или он найдет лазейки, которые позволят выполнять на вашем компьютере чужой код с правами администратора. Так хакер получает неограниченную власть над вашим компьютером, может устанавливать на нем свои программы и запускать их.

Возможно, вы спросите, как злоумышленник может получить контроль над вашими программами. Для этого практически всегда используются ошибки, содержащиеся в программном коде. Например, при пересылке данных искусственно выполняется так называемое переполнение буфера. Эта ошибка, в свою очередь, применяется для того, чтобы добавить в программу чужой код и выполнить его. Конечно, есть и другие методы, но все они связаны с поиском и использованием

ошибок в программе, снижающих ее надежность и позволяющих использовать ее не по назначению либо манипулировать ею.

Меры предосторожности. Идеальных программ, совсем не содержащих ошибок, нет и в ближайшем будущем не будет (здесь я не говорю о простейших и самых миниатюрных программах). По этой причине с течением времени были разработаны всевозможные методы, позволяющие свести к минимуму риски, которые связаны с программными ошибками. К наиболее проверенным мерам предосторожности относится отказ от использования демонов с правами администратора. Следует также устанавливать как можно меньше программ или сценариев с битом `setuid`, не позволяющих выполнять код в стеке (для этих целей в Red Hat было разработано расширение ядра Exec Shield) и т. д.

Основы SELinux. Программой следующего поколения является расширение ядра SELinux, которое первоначально было разработано Агентством национальной безопасности США как свободная программа. Работа этого расширения заключается в том, что SELinux, основываясь на заданных правилах, наблюдает за выполнением программ. Этот метод называется *обязательным контролем доступа* (Mandatory Access Control, MAC). Если правило нарушается, то SELinux запрещает выполнять операцию или регистрирует предупреждение. При этом расширение SELinux не завершает работу программы, нарушающей правила. От программы зависит, как она отреагирует на ситуацию, в которой не сможет обратиться к определенному файлу или воспользоваться сетевым интерфейсом.

Правила MAC обеспечивают значительно более тщательный контроль безопасности, чем система прав доступа UNIX. С помощью MAC программе можно запретить доступ к определенным каталогам и сетевым функциям, независимо от прав доступа или учетных записей, с которыми она работает. Поскольку эти правила отслеживаются на уровне ядра, они действуют и в том случае, когда программа выходит из-под контроля из-за ошибки или недоработки в области безопасности.

Агентство национальной безопасности США — это информационная служба, которая обеспечивает в том числе надзор за обменом электронной информацией и ее шифрование. Итак, хотя SELinux в определенной степени и происходит из сфер «тайной полиции», в дополнении Linux такими функциями нет ничего предосудительного — код SELinux открытый, контролируется многочисленными независимыми экспертами и официально входит в состав ядра версии 2.6 и выше.

Правила SELinux. Без необходимых правил расширение SELinux бесполезно. Таким образом, от качества правил напрямую зависит, насколько надежнее станет система, оборудованная SELinux. Из распространенных дистрибутивов пока только Red Hat вложил достаточно времени и усилий в разработку таких правил. Все испытания проводятся в системе Fedora, которая в данном случае играет роль экспериментального полигона. Правила, которые будут признаны практичными в Fedora, входят в состав новых версий Red Hat Enterprise.

В этом разделе расширение SELinux описано на базе Fedora 11. Насколько широко SELinux закрепится за пределами Red Hat и Fedora, предположить сложно. В Novell и SUSE, а также в Ubuntu используется альтернативное решение AppArmor — о нем речь пойдет в следующем разделе. Правда, не так давно SUSE свернула программу разработки AppArmor и интегрировала SELinux в дистрибутив настолько глубоко, что работать с этим расширением в принципе возможно и без

перекомпиляции ядра. Остается лишь гадать, является ли этот шаг началом конца AppArmor.

Критика SELinux. Расширение SELinux небезупречно. Два основных момента, из-за которых оно подвергается критике, таковы.

- Файлы нужно помечать расширенными атрибутами, чтобы обеспечить их правильное взаимодействие с SELinux. Для этого требуются файловые системы, поддерживающие расширенные атрибуты (NFS к их числу не относится!). Кроме того, возникают проблемы с обновлениями и резервным копированием.
- Основная проблема SELinux — исключительная сложность. Для одного только обеспечения безопасности основных сетевых служб требуются тысячи правил. Лишь немногие эксперты способны судить об эффективности этих правил. По существующим наборам правил не хватает документации. По этим причинам среднестатистический пользователь Linux обычно оказывается не в состоянии приспособить SELinux к своим нуждам.

  Расширение SELinux по праву снискало среди большинства своих пользователей славу «черного ящика». Из-за сложности программы ее установка обычно не обходится без ошибок. В результате при возникновении первых же нестыковок возникает соблазн просто выключить компьютер, а потом начать все заново.

Ссылки. SELinux более подробно описывается на следующих сайтах:

- http://www.nsa.gov/research/selinux/;
- http://sourceforge.net/projects/selinux/;
- http://www.crypt.gen.nz/selinux/faq.html;
- http://fedoraproject.org/wiki/SELinux.

Альтернативы. Наиболее популярной альтернативой SELinux является система, применяемая в SUSE и Ubuntu AppArmor (см. раздел 29.2). Кроме того, в версии ядра 2.6.25 появилась еще одна система MAC под названием Smack (http://lwn.net/Articles/252562/). Но пока еще рано судить, сможет ли Smack реально конкурировать с SELinux и AppArmor.

Внутренняя организация и принцип работы SELinux

SELinux базируется на двух основных предпосылках: во-первых, это правильное обозначение всех файлов и процессов с помощью так называемого контекста защиты, во-вторых, это правила, которые должны выполняться наблюдаемыми процессами.

Контекст защиты

Принцип работы SELinux основан на том, что любой объект (например, файл) и любой субъект (допустим, процесс) связаны с контекстом защиты. С файлами контекст защиты сохраняется в виде расширенных атрибутов. В таком случае информация о защите непосредственно связана с файлом и не зависит от имени файла. Контекст защиты определенного файла проще всего узнать с помощью команды `ls -Z`. В качестве альтернативы можно попробовать `etfattr -n security.selinux -d` *`имя_файла`*.

```
user$ ls -Z /usr/sbin/httpd
-rwxr-xr-x root root system_u:object_r:httpd_exec_t:s0 /usr/sbin/httpd
user$ ls -Z /etc/httpd/conf/httpd.conf
-rw-r--r-- root root system_u:object_r:httpd_config_t:s0 /etc/httpd/conf/httpd.conf
user$ getfattr -n security.selinux -d /etc/httpd/conf/httpd.conf
getfattr: Removing leading '/' from absolute path names
# file: etc/httpd/conf/httpd.conf
security.selinux="system_u:object_r:httpd_config_t:s0\000"
```

Для работы SELinux необходимо, чтобы со всеми файлами был сохранен правильный контекст защиты. Чтобы этот контекст работал и тогда, когда вы уже после установки будете создавать новые файлы, во многих каталогах существуют правила SELinux, которые автоматически присваивают нужные контексты файлам, сохраняемым в этих каталогах. Если данный механизм не работает автоматически, то вы можете сами настроить правильный контекст с помощью команды `chcon`:

```
user$ chcon user_u:object_r:user_home_t тестовый_файл
```

При работе с процессами вместо слова «контекст» часто применяется термин «домен». Чтобы определить контекст (домен) того или иного процесса, используйте команду `ps axZ`. Как правило, процесс принимает контекст той учетной записи, с которой он был запущен. Но контекст также можно автоматически изменить сразу после запуска с помощью правила SELinux. Это бывает необходимо в тех случаях, когда определенная программа (например, Firefox) должна получать заданный контекст независимо от того, какая учетная запись ее запустила.

```
user$ ps axZ | grep httpd
user_u:system_r:httpd_t:s0  3758 ?   Ss    0:00 /usr/sbin/httpd
user_u:system_r:httpd_t=s0  3760 ?   S     0:00 /usr/sbin/httpd
...
```

Контекст защиты состоит из трех или четырех частей, разделяемых двоеточиями:

```
пользователь:роль:тип:mls-компонент
```

Самая важная часть — третья, в которой указывается тип файла или процесса. Большинство правил SELinux интерпретируют эту информацию. Подробное описание всех четырех частей контекста защиты находится по адресу http://fedoraproject.org/wiki/SELinux/SecurityContext.

Правила

В обобщенном виде контекст правил SELinux выглядит так:

```
allow тип1_t тип2_t:класс { операции };
```

Приведу пример: следующее правило разрешает процессам, имеющим тип контекста `httpd_t`, создавать новые файлы в каталогах с типом контекста `httpd_log_t`.

```
allow httpd_t httpd_log_t:dir create;
```

Как правило, в отдельно взятой конфигурации SELinux таких правил десятки тысяч! Чтобы система работала быстрее, SELinux обычно ожидает правил не в виде

текста, а в двоичной форме. Проводя аналогию с программированием, правила можно сравнить с результатом компилирования. Общее описание этапов, которые необходимо выполнить, чтобы добавить к существующей конфигурации SELinux собственный модуль правил, дается в часто задаваемых вопросах по SELinux: http://docs.fedoraproject.org/en-US/Fedora/13/html/SELinux_FAQ/.

Со временем сформировались некоторые устойчивые наборы правил, предназначенные для достижения определенных целей.

- Strict — в Fedora 2 набор правил strict был активен и содержал правила для выполнения всех процессов, но создавал, таким образом, больше проблем, чем решал.
- Targeted — в Fedora 3 и выше по умолчанию стал использоваться набор правил targeted. Его правила защищают только специально выбранные сетевые службы.

  К сожалению, очень плохо документирован механизм, в соответствии с которым осуществляется защита, а также не описано, какую контекстную информацию должны иметь данные и какие управляющие параметры существуют (см. ниже). Для некоторых важнейших служб в справке `man` есть отдельные страницы: `ftpd_selinux`, `httpd_selinux`, `named_selinux`, `nfs_selinux`, `samba_selinux` и т. д. Полный список выдает следующая команда:

  ```
  user$ rpm -qd selinux-policy | grep man8
  ```

- MLS — в качестве альтернативы можно использовать разработанный для сервера набор правил MLS (Multilevel Security, многоуровневая безопасность) (пакет `selinux-policy-mls`). Этот набор был разработан для того, чтобы RHEL мог получить сертификат класса EAL 4. Такой сертификат в США требуется при выпуске определенных (часто — военных) приложений, хотя фактическое улучшение безопасности при этом не так и велико. Более подробная информация приводится на следующих сайтах:
  - http://fedoraproject.org/wiki/SELinux/FedoraMLSHowto;
  - http://en.wikipedia.org/wiki/Common_Criteria;
  - http://en.wikipedia.org/wiki/Evaluation_Assurance_Level.

При разработке наборов правил SELinux используется механизм Reference Policy, созданный компанией Tresys (http://oss.tresys.com/projects/refpolicy). Он очень удобно применяется в среде разработки. Все действующие наборы правил SELinux были созданы с помощью этого механизма.

Параметры SELinux (булевы)

Итак, вы уже понимаете, что вносить изменения в наборы правил достаточно сложно. Чтобы у нас было определенное поле для маневра, не требующее изменения правил, в наборе `targeted` предусмотрены булевы параметры, которые можно изменять на ходу. В Fedora/Red Hat для этого обычно используется графический пользовательский интерфейс `system-config-selinux` (рис. 29.1). Эта программа находится в пакете `policycoreutils-gui`, который по умолчанию не устанавливается. В качестве альтернативы можно попробовать команду `getsebool`, узнающую значения булевых параметров; `setsebool` предназначена для изменения таких параметров.

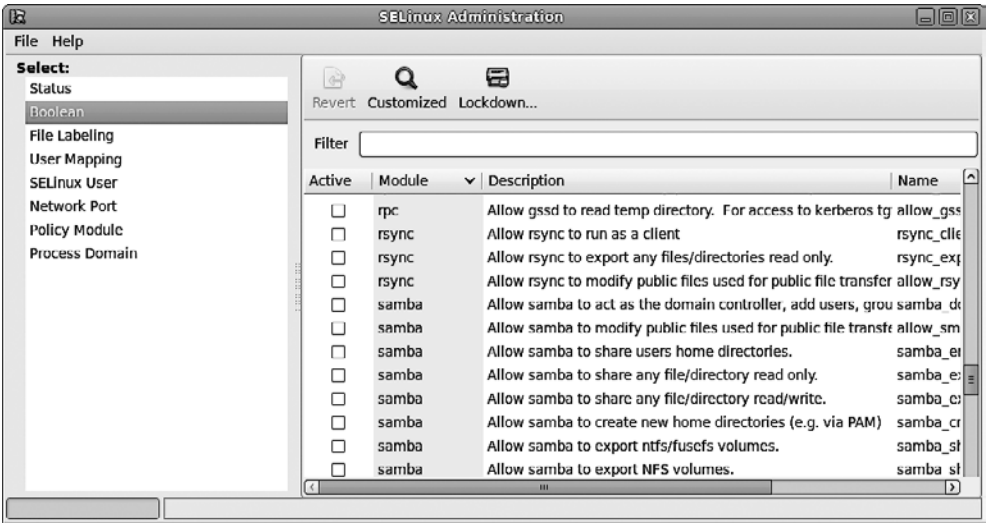


Рис. 29.1. Изменение параметров SELinux

Запуск и конфигурация

SELinux входит в состав ядра, поэтому специально запускать программу (например, через систему Init-V) не требуется. Необходимость в демоне SELinux или других фоновых процессах также отпадает.

Конфигурация производится в файлах, расположенных в каталоге `/etc/selinux`. Определяющее значение имеет файл `/etc/selinux/config`. В нем указывается, в каком режиме работает SELinux (`enforcing`, `permissive` или `disabled`) и какой набор правил действует (`strict` или `targeted`). Изменения, внесенные в этот файл, вступают в силу только после перезапуска.

```
# /etc/selinux/config
SELINUX=enforcing
SELINUXTYPE=targeted
```

Статус

Команда `setstatus` определяет текущий статус SELinux. На моем испытательном компьютере, где SELinux работает в режиме `targeted`, получены такие результаты:

```
root# sestatus
SELinux status:                 enabled
SELinuxfs mount:                /selinux
Current mode:                   enforcing
Mode from config file:          enforcing
Policy version:                 24
Policy from config file:        targeted
```

Устранение проблем, связанных с SELinux

Рассмотрим, какие возможности реагирования на нарушение правил SELinux существуют:

- в наборе правил находится параметр, который может быть причиной проблемы, а затем его настройки корректируются с помощью `system-config-security`;
- изменяется контекстная информация файла, вызывающего проблемы;
- набор правил изменяется или дополняется; для этого требуются глубокие знания SELinux, и информации, сообщаемой в этом разделе, недостаточно;
- SELinux просто отключается.

Проблемы, вызываемые SELinux, не всегда очевидны. Например, если скопировать дерево каталогов, содержащее HTML-файлы, из каталога NFS в каталог `/var/www/html` командой `cp -a`, то Apache больше не сможет прочитать эти файлы. Причина заключается в том, что при использовании параметра `cp -a` вместе с файлами копируются и расширенные атрибуты, и контекстная информация SELinux. Из-за этого файлы, скопированные в `/var/www/html`, не могут автоматически получить нужную контекстную информацию в соответствии с правилом SELinux. Контекстная информация HTML-файлов в таком случае имеет вид `system_u:object_r:nfs_t`, в то время как правильный вариант — `user_u:object_r:httpd_sys_content_t`. Чтобы избежать подобных проблем, измените `cp -a` на `cp -r`.

Сам Apache «ничего не знает» о существовании SELinux. Программа просто обнаруживает, что не может прочитать некоторые файлы, и выдает непонятное сообщение об ошибке You don't have permission to access <filename> (У вас нет прав на доступ к <имя файла>). Только взглянув в `/var/log/messages`, вы поймете, что проблемы с доступом возникли из-за SELinux.

```
root# less /var/log/messages
...
Aug 31 12:45:45 mars setroubleshoot: SELinux is preventing the
  httpd from using potentially mislabeled files (test.html).
  For complete SELinux messages run sealert -l 2f35f9a0-bd1b-40ab-9779-3a640b99ef10
```

Если теперь вы выполните `sealert`, как указано в файле протокола, на экран будет выведен почти 100-строчный текст, важнейшая информация из которого изложена здесь.

```
root# sealert -l dccb472d-6dd8-49d2-b7d7-2658e082c805
SELinux закрывает для /usr/sbin/httpd доступ к чтению файла index.html.

***** Plugin catchall_boolean (89.3 confidence) suggests ******************
Если вы хотите разрешить httpd читать пользовательский контент, то сообщите SELinux
об этом, активизировав 'httpd_read_user_content'. Можете почитать страницу справки
'user_selinux', на которой изложена более подробная информация. Выполните setsebool -P
httpd_read_user_content 1

***** Plugin catchall (11.6 confidence) suggests **************************
Если вы считаете, что нужно по умолчанию разрешить httpd доступ для чтения к файлу
index.html, то должны пометить это как сообщение об ошибке. Чтобы разрешить такой
доступ, можно создать локальный модуль с регулирующими правилами. Чтобы разрешить
доступ прямо сейчас, выполните следующие команды:
# grep httpd /var/log/audit/audit.log | audit2allow -M mypol
# semodule -i mypol.pp
...
```

Мало того, что качество текста оставляет желать лучшего — так и в содержательном отношении предложенные решения совершенно неприменимы. Чтобы действительно устранить проблему, нужно правильно настроить контекстную информацию интересующих вас файлов с помощью `restorecon`:

```
root# restorecon -R -v /var/www/html/*
```

Отключение SELinux

Если требуется активизировать SELinux лишь ненадолго, то запустите `system-config-security` и включите режим `Permissive`. Расширение SELinux продолжит работу и будет регистрировать нарушения правил в `/var/log/messages`. Но в таком режиме SELinux допускает ошибки и не блокирует программы, которые их совершают. Аналогичное действие оказывает команда `setenforce 0`.

Разумеется, SELinux в `system-config-security` можно полностью отключить (настройка `Disabled`). Но это стоит делать лишь в тех случаях, когда вы больше не планируете работать с этим расширением. Причина заключается в том, что как только вы отключаете SELinux, все его правила больше не смогут присваивать контекстную информацию новым файлам. Если позже снова включить SELinux, то файлы, в которых нет контекстной информации, будут вызывать проблемы. При последующей корректировке контекстных данных вам поможет команда `restorecon`, но подобный процесс утомителен и чреват ошибками.

Если SELinux будет вызывать проблемы уже при старте и мешать запускать компьютер, то в ядре нужно установить параметр `selinux=0`, не позволяющий системе SELinux запускаться. Активизация происходит только после перезапуска системы. Можно также применить параметр `enforcing=0`. В таком случае SELinux запускается, а нарушения правил не регистрируются.

29.2. AppArmor

Чтобы не адаптировать к своему дистрибутиву сложную систему SELinux, Novell в 2005 году приобрела фирму Immunix и переименовала ее программу Subdomain (предназначенную для обеспечения безопасности) в AppArmor. Компания Novell привела AppArmor в соответствие с лицензией GPL и разработала несколько модулей YaST для администрирования. Наконец, AppArmor получил достойное признание в среде разработчиков ядра Linux и был официально включен в код версии ядра 2.6.36. AppArmor применяется в дистрибутивах SUSE и Ubuntu.

Осенью 2007 года компания Novell уволила разработчиков AppArmor. На сайте Novell (openSUSE) с тех пор относительно AppArmor царит тишина. AppArmor, конечно, применяется и дальше, но с тех пор не произошло заметных изменений ни в правилах, ни в качестве инструментов администрирования. Дальнейшей разработкой AppArmor с тех пор занимаются, в основном, разработчики, занятые в Canonical.

Как и SELinux, AppArmor является системой безопасности типа MAC (мандатного управления доступом). В отличие от SELinux, правила AppArmor основаны на абсолютных именах файлов. Поэтому не требуется специально обозначать все файлы с помощью расширенных атрибутов (EA). Более того, AppArmor работает

и в тех файловых системах, которые не поддерживают EA. В правилах AppArmor можно применять джокерные символы. По этой причине в типичных практических ситуациях для работы с AppArmor требуется определять значительно меньше правил, чем с SELinux. Отсюда проистекает еще одно значительное преимущество AppArmor перед SELinux: программа построена гораздо проще.

Естественно, у AppArmor есть и недостатки.

- Эксперты по безопасности из Red Hat придерживаются мнения, что при указании в правилах абсолютных путей вы постоянно подвергаете систему риску (защиту AppArmor можно обойти, переименовав файлы и каталоги; разумеется, для этого агрессор должен иметь достаточно широкие права).
- Набор правил AppArmor не так масштабен, как в SELinux. По умолчанию защита распространяется на меньшее количество программ. Хотя пользователю и проще самостоятельно создавать и изменять правила, такая «самодельная» безопасность производит впечатление некоторого непрофессионализма.

Ссылки. В этом разделе сделано лишь введение в AppArmor. Более подробная информация приводится на следующих сайтах:

- http://www.novell.com/documentation/apparmor/ (старые пособия в формате PDF);
- https://help.ubuntu.com/community/AppArmor;
- https://wiki.ubuntu.com/SecurityTeam/KnowledgeBase/AppArmorProfiles.

AppArmor в Ubuntu

В следующих моделях рассмотрен вариант AppArmor, поставляемый с Ubuntu 12.04. Замечания, касающиеся SUSE, даются в подразделе «AppArmor в SUSE» далее.

В Ubuntu AppArmor по умолчанию интегрирована в ядро. Система безопасности запускается в начале процесса Init-V сценарием `/etc/init.d/boot.apparmor`, который обращается к функциям, определенным в `/lib/apparmor/rc.apparmor.functions`. Сценарий `boot.apparmor` считывает базовую конфигурацию из каталога `/etc/apparmor` и загружает все данные правил в каталог `/etc/apparmor.d`.

Сценарий Init-V пользуется функциями, определенными в файле `/etc/apparmor/rc.apparmor.functions`.

Команда `aa-status` дает обзор актуального состояния AppArmor. Эта команда выводит как список всех профилей, так и список процессов, отслеживаемых в настоящий момент. В приведенном ниже списке показано, какие службы корневого сервера Ubuntu отслеживаются в настоящий момент.

```
root# aa-status
apparmor module is loaded.
6 profiles are loaded.
6 profiles are in enforce mode.
   /sbin/dhclient
   /usr/lib/NetworkManager/nm-dhcp-client.action
   /usr/lib/connman/scripts/dhclient-script
   /usr/sbin/mysqld
   /usr/sbin/ntpd
   /usr/sbin/tcpdump
```

```
0 profiles are in complain mode.
2 processes have profiles defined.
2 processes are in enforce mode.
   /usr/sbin/mysqld (851)
   /usr/sbin/ntpd (14794)
0 processes are in complain mode.
0 processes are unconfined but have a profile defined.
```

При запуске AppArmor файловая система `securityfs` подключается в каталоге `/sys/kernel/security`. Файлы из этого каталога содержат информацию об активных профилях, количестве обнаруженных нарушений правил и т. д.

Правила (профили)

Действие AppArmor зависит от правил отслеживания. Эти правила также называются профилями и находятся в файлах каталога `/etc/apparmor.d/`.

Официально поддерживаемые профили правил обычно предоставляются в одноименных пакетах. Поэтому сразу после установки Ubuntu `/etc/apparmor.d` почти пуст и заполняется по мере установки серверных служб.

Кроме того, можно установить пакет `apparmor-profiles` из репозитория пакетов `universe`. В нем содержатся многочисленные дополнительные профили, которые, правда, официально не поддерживаются. Большинство профилей работает в так называемом complain-режиме. В этом режиме нарушения правил протоколируются, но не предотвращаются принудительно. С помощью команд `aa-enforce` и `aa-complain` можно изменять режим профиля. Обеим командам нужно передавать полный путь к той программе, которую планируется отслеживать.

```
root# aa-enforce /usr/sbin/dnsmasq
Setting /usr/sbin/dnsmasq to enforce mode.
root# aa-complain /usr/sbin/dnsmasq
Setting /usr/sbin/dnsmasq to complain mode.
```

В качестве альтернативы командам `aa-enforce` и `aa-complain` можно передавать имена файлов профилей. Таким образом, вы сможете с легкостью одновременно менять режим нескольких профилей:

```
root# cd /etc/apparmor.d
root# aa-enforce usr.lib.dovecot*
```

При работе с серверными службами после активизации профиля AppArmor нужно также перезапустить соответствующую программу:

```
root# service <name> restart
```

В отрыве от режима AppArmor профили, конечно, имеют значение лишь тогда, когда соответствующий сетевой демон или команда действительно выполняется. Чтобы узнать, какие программы в настоящее время находятся под наблюдением, используется вышеупомянутая команда `aa-status`.

Пользователи Ubuntu, любящие заниматься экспериментами, могут испробовать тестовые профили, которые расположены в каталоге `/usr/share/doc/apparmorprofiles/extras/`. Советы по установке содержатся в файле `README`, находящемся в этом же каталоге. Если профили, действующие в режиме `complain`, регистрируют нарушение

правил, хотя программа и работает правильно, нужно расширить соответствующий профиль. В этом вам поможет команда `aa-logprof` (см. `man aa-logprof`). Если через несколько недель тестового использования у вас сложится впечатление, что профили работают нормально, то рекомендуется активизировать режим `enforce`.

Построение файлов правил

Файлы правил, называемые в AppArmor Profile (профиль), записываются в легко интерпретируемом текстовом формате. В следующих строках приведен пример правил AppArmor для `named`:

```
#include <tunables/global>

/usr/sbin/named {
   #include <abstractions/base>
   #include <abstractions/nameservice>

   capability net_bind_service,
   capability setgid,
   capability setuid,
   capability sys_chroot,
   capability sys_resource,

   # /etc/bind должен быть только для чтения bind.
   # /var/lib/bind для динамически обновляемых файлов зон и файлов журнала.
   # /var/cache/bind для ведомых данных/данныхзаглушки,
   # если не мы являемся авторами этих данных.
   # См. /usr/share/doc/bind9/README.Debian.gz
   /etc/bind/** r,
   /var/lib/bind/** rw,
   /var/lib/bind/ rw,
   /var/cache/bind/** rw,
   /var/cache/bind/ rw,

   # gssapi
   /etc/krb5.keytab kr,
   /etc/bind/krb5.keytab kr,

   # ssl
   /etc/ssl/openssl.cnf r,

   # Пакет dnscvsutil
   /var/lib/dnscvsutil/compiled/** rw,

   /proc/net/if_inet6 r,
   /proc/*/net/if_inet6 r,
   /usr/sbin/named mr,
   /var/run/named/named.pid w,
   /var/run/named/session.key w,
   # поддержка resolvconf
   /var/run/named/named.options r,
```

```
    # Некоторые специалисты предпочитают записывать логи в /var/log/named/,
    # а не перекладывают выполнение сложных задач на syslog
    /var/log/named/** rw,
    /var/log/named/ rw,
}
```

Сначала в файлах правил считываются некоторые включаемые (include-) файлы, а потом определяются базовые признаки программы (см. справку `man capabilities`). Остальные правила регламентируют, какие файлы программа может использовать и как.

В файлах правил AppArmor джокерный символ * применяется в качестве подстановочного для любого количества символов. ** имеет подобное значение, но может включать, в том числе, символ /. В таком случае, он охватывает и файлы из всех подкаталогов. Права доступа выражаются буквами и сочетаниями букв. В табл. 29.1 приведены важнейшие из них. Комбинации `?x` управляют правами подпроцессов, запускаемых основной программой.

Таблица 29.1. Простейшие права доступа AppArmor

Сокращение	Значение
r	Открывает доступ для чтения (read)
w	Открывает доступ для изменения (write)
a	Позволяет дополнять файл (append)
l	Применяет к жестким ссылкам те же правила, что и к исходному файлу (link)
k	Позволяет блокировать файл (lock)
m	Допускает исполнение функций mmap (разрешает исполняемое отображение)
ix	Программа наследует правила базовой программы (inherent execute)
px	Программа имеет собственный профиль AppArmor (discrete profile execute)
ux	Выполнение программы без правил AppArmor (unconstrained execute)

Параметры правил (tunables)

В AppArmor предусмотрен механизм, позволяющий без труда изменять некоторые параметры правил. Эти параметры определяются в файлах каталога `/etc/apparmor.d/tunables`. Но в актуальной реализации присутствует всего три параметра, позволяющих индивидуально настраивать расположение домашнего каталога и каталога `/proc`. Если на вашем сервере предусмотрены и другие места для размещения домашних каталогов, кроме `/home`, измените переменную `@{HOMEDIRS}`.

```
# Файл /etc/apparmor.d/tunables/home
@{HOME}=@{HOMEDIRS}/*/ /root/
@{HOMEDIRS}=/home/ /home1/ /myhome/

# Файл /etc/apparmor.d/tunables/proc
@{PROC}=/proc/
```

Логирование и техническая поддержка

Подробное описание зарегистрированных нарушений правил, произошедших в режимах `complain` или `enforce`, передаются в форме сообщений ядра и по умолча-

нию перечисляются в файлах /var/log/kern.log и /var/log/syslog. Сообщения AppArmor легко опознать по ключевому слову audit.

```
root# grep audit /var/log/kern.log
[...] audit(1238580174.435:3): type=1503 operation="inode_permission"
      requested_mask="a::" denied_mask="a::" name="/dev/tty"
      pid=6345 profile="/usr/sbin/cupsd" namespace="default"
[...] audit(1238580174.435:4): type=1503 operation="inode_permission"
      requested_mask="w::" denied_mask="w::" name="/etc/krb5.conf"
      pid=6345 profile="/usr/sbin/cupsd" namespace="default"
```

В 99 % случаев audit-сообщения являются признаком того, что правила AppArmor неполны. Возможно, конечно, что сама программа работает с ошибками, но это маловероятно. Дать уверенное заключение могут только эксперты по конкретной программе. Как правило, сложно полностью адекватно отреагировать на нарушение правил.

Если вы полагаете, что интересующая вас программа работает правильно, то нужно переключить модуль в complain-режим и отметить audit-сообщение в системе регистрации багов Ubuntu (https://bugs.launchpad.net/). Кроме того, можно попытаться дополнить профиль новым правилом, обеспечивающим необходимый приоритет. Или просто проигнорируйте сообщение, если программа продолжает работать без проблем. Вы как администратор сервера редко сможете улучить минуту, чтобы подробно разобраться в деталях правил AppArmor.

AppArmor в SUSE

Конфигурация и запуск

В openSUSE 12.1 и выше AppArmor больше не устанавливается по умолчанию. Если вы хотите работать с AppArmor, то сначала нужно установить соответствующие пакеты. Проще всего для этого активизировать в YaST-модуле Установка и удаление программ схему Novell AppArmor.

AppArmor запускается в начале процесса Init-V сценарием /etc/init.d/boot.apparmor. Этот сценарий обращается к функциям, определяемым в файле /lib/apparmor/rc.apparmor.functions. Команда boot.apparmor считывает базовую конфигурацию из каталога /etc/apparmor и загружает все файлы правил из каталога /etc/apparmor.d.

В отличие от SELinux, в AppArmor пока разработаны надежные защитные профили для небольшого количества программ. Отдельные профили, требующие доработки, находятся в каталоге /etc/apparmor/profiles/extras. Однако они уже много лет имеют статус экспериментальных и по умолчанию не активизируются.

Изменения, внесенные в конфигурацию, вступят в силу только после того, как вы заново запустите AppArmor или считаете файлы правил:

```
root# /etc/init.d/boot.apparmor restart  (Перезапустить AppArmor)
root# /etc/init.d/boot.apparmor reload   (Заново загрузить правила AppArmor)
```

Модули YaST

Для управления AppArmor в конфигурационной программе YaST дистрибутива SUSE предусмотрены различные модули (рис. 29.2). В контрольной панели

AppArmor программу можно (де)активировать либо выбрать для отдельных профилей режим `enforce` или `complain`. В режиме `complain` нарушения правил регистрируются, но работа «ошибающейся» программы не прекращается.

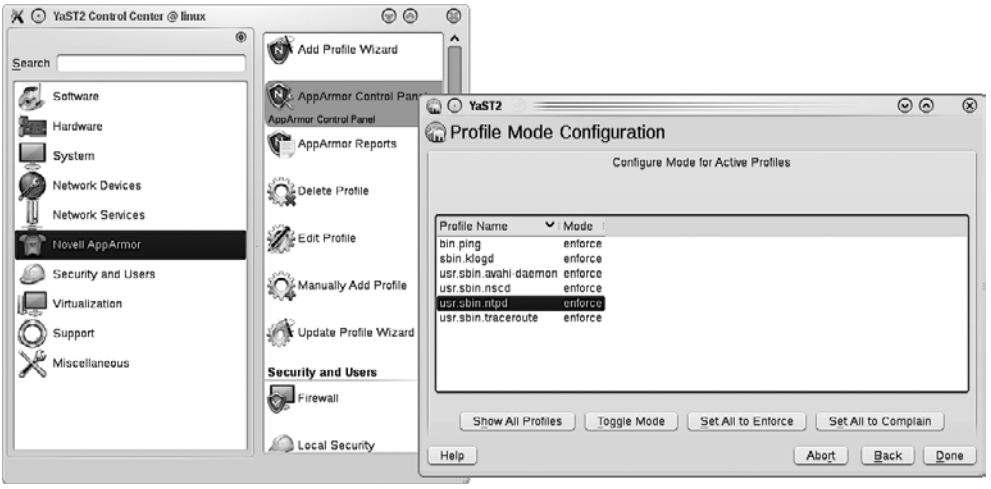


Рис. 29.2. Конфигурация AppArmor с помощью YaST

В модуле Edit Profile (Редактировать профиль) можно изменять имеющийся профиль. Но, как правило, лучше обрабатывать файл правил в обычном текстовом редакторе. При составлении правил модуль YaST предоставляет только некоторые вспомогательные функции.

Гораздо интереснее модуль Add Profile Wizard (Ассистент для добавления профилей): с его помощью вы можете в своеобразном учебном режиме сами разработать защитный профиль для программы. Для этого приложение запускается, а потом вы пробуете как можно больше ее функций. На этом этапе программа еще не защищена AppArmor, поэтому необходимо полностью исключить возможность атаки во время такого обучения. Ассистент регистрирует все обращения программы к файлам и предлагает правила, подходящие для защиты программы. Теперь вы должны утвердить или изменить отдельные правила. Полученный в результате профиль будет сохранен под именем `/etc/apparmor.d/programmname`.

С помощью ассистента вам будет достаточно просто создавать собственные защитные профили. Однако полученные таким образом профили скорее всего будут неидеальны: с одной стороны, работая с программой, вы вполне можете не проверить некоторые ее функции, а для них будут нужны дополнительные правила. С другой — автоматическое генерирование правил безопасности говорит о невысоком качестве таких правил. Таким образом, после работы ассистента вам предстоит вручную внести еще некоторые доработки профиля безопасности.

Если с имеющимся профилем возникают проблемы, вы можете использовать Update Profile Wizard (Ассистент для обновления профилей), чтобы дополнить профиль новыми правилами. Ассистент изучает файлы AppArmor, в которые занесена информация о возникших ошибках, и вырабатывает новые правила на базе этой информации.