Скотт Граннеман

НЕОБХОДИМЫЙ КОД И КОМАНДЫ

Linux®

КАРМАННЫЙ СПРАВОЧНИК

Linux®

PHRASEBOOK

ESSENTIAL CODE AND COMMANDS

Scott Granneman

DEVELOPER'S LIBRARY

Sams Publishing, 800 East 96th Street, Indianapolis, Indiana 46240 USA

Linux®

КАРМАННЫЙ СПРАВОЧНИК

НЕОБХОДИМЫЙ КОД И КОМАНДЫ

Скотт Граннеман

Москва • Санкт-Петербург • Киев
2010

ББК 32.973.26-018.2.75
Г77
УДК 004.451.47

Издательский дом "Вильямс"
Зав. редакцией *С. Н. Тригуб*
Перевод с английского и редакция *В. В. Вейтмана*
По общим вопросам обращайтесь
в Издательский дом "Вильямс" по адресу:
info@williamspublishing.com, http://www.williamspublishing.com

Граннеман, Скотт.

Г77 Linux. Карманный справочник. : Пер. с англ. — М. : ООО "И.Д. Вильямс", 2010. — 416 с. : ил. — Парал. тит. англ.

ISBN 978-5-8459-1118-6 (рус.)

Данная книга представляет собой краткое пособие по основным командам операционной системы Linux. В первых главах представлены самые элементарные сведения о работе с системой. По мере чтения книги материал усложняется; освоив его, читатель сможет решать достаточно серьезные задачи. Начинающие пользователи, только приступающие к изучению Linux, найдут сведения о самых необходимых им командах. Но предполагаемая аудитория не ограничивается новичками. Материал книги также напомнит опытным пользователям команды и опции, которые они успели забыть или которым они по каким-то причинам ранее не уделяли внимания.

ББК 32.973.26-018.2.75

ISBN 978-5-8459-1118-6 (рус.)
ISBN 0-672-32838-0 (англ.)

Оглавление

Содержание

Об авторе

Скотт Граннеман — преподаватель, консультант и автор нескольких книг. Его интересы лежат в области программного обеспечения с открытым исходным кодом, что демонстрируют его первые две книги: *Don't Click on the Blue E!: Switching to Firefox* и *Hacking Knoppix*, а также личного вклада в *Ubuntu Hacks*. Граннерман также ведет ежемесячную рубрику в SecurityFocus, материалы которой в основном посвящены общим вопросам безопасности. Кроме того, его работы публикуются в *Linux Magazine* и *The Open Source Weblog*.

Будучи адъюнкт-профессором Вашингтонского университета, Скотт читает различные курсы по информационным технологиям и работе с глобальной сетью. Его лекции посещают люди разного возраста — вчерашние школьники и солидные дипломированные специалисты. В последние десять лет акцент лекций был заметно смещен в сторону Linux и других технологий с открытым исходным кодом. Скотт старается донести до слушателей знания о новых направлениях в разработке программ, обеспечивающих широкие возможности.

Будучи администратором WebSanity — компании, занимающейся созданием Web-узлов и хостингом, — он работает с коммерческими и неприбыльными организациями и помогает им использовать Интернет для обеспечения взаимодействия, расширения объема продаж и предоставляемых услуг. Он управляет вычислительными средствами фирмы, работающими на базе Unix, воплощая на практике те знания, которыми он делится со слушателями на лекциях и описывает в своих книгах. Скотт тесно взаимодействует с партнерами и разработчиками системы управления содержимым (Content Management System — CMS), используемой в WebSanity.

Посвящается пользователям Linux всех поколений.

Благодарности

Никто не способен изучить оболочку Linux без чьей-либо помощи. Моими учителями были сотни специалистов, десятилетиями работавшие с системой Unix. В своих книгах они рассказали мне и другим читателям о невероятных возможностях, предоставляемых командной строкой Linux. Книги, Web-узлы, блоги и статьи в журналах помогли мне изучить оболочку bash и помогают в этом по сей день. Если я смогу донести до своих читателей хоть малую часть того, что смог получить сам, я буду считать свою задачу выполненной.

Помимо моих учителей, я хочу поблагодарить тех, кто непосредственно помогал мне в работе над этой книгой.

Огромную помощь постоянно оказывала мой агент, Лаура Левин (Laura Lewin).

Мои редакторы не только с пониманием относились к моей работе, но и находили способы стимулировать ее.

Неоценимую помощь оказал Роберт Сайтек (Robert Citek), большой знаток Linux. Когда у меня возникали вопросы, у него всегда находились ответы на них.

Мой давний друг и деловой партнер Дженс Картон (Jans Carton) всегда помогал мне сосредоточиться на конкретных вопросах и был неисчерпаемым источником информации о многих новых командах и опциях. Думал ли я, когда мы познакомились в пятом классе школы, что он станет таким выдающимся специалистом?

Джерри Брайан (Jerry Bryan) внимательно читал написанный мною текст и исправлял все грамматические ошибки. Благодаря Джерри я научился более или менее грамотно излагать свои мысли на бумаге.

Моя жена, Дениза Либерман (Denise Lieberman), терпеливо выслушивала мои бессвязные восклицания, когда мне приходила в голову новая идея. Спасибо, Денизаl

И наконец, у меня остались самые теплые воспоминания о том, как моя собака Либби в самый разгар работы клала мне лапы на колени и требовала, чтобы я погладил ее.

От издательства

Вы, читатель этой книги, и есть главный ее критик и комментатор. Мы ценим ваше мнение и хотим знать, что было сделано нами правильно, что можно было сделать лучше и что еще вы хотели бы увидеть изданным нами. Нам интересно услышать и любые другие замечания, которые вам хотелось бы высказать в наш адрес.

Мы ждем ваших комментариев и надеемся на них. Вы можете прислать нам бумажное или электронное письмо либо просто посетить наш Web-сервер и оставить свои замечания там. Одним словом, любым удобным для вас способом дайте нам знать, нравится или нет вам эта книга, а также выскажите свое мнение о том, как сделать наши книги более интересными для вас.

Посылая письмо или сообщение, не забудьте указать название книги и ее авторов, а также ваш обратный адрес. Мы внимательно ознакомимся с вашим мнением и обязательно учтем его при отборе и подготовке к изданию последующих книг. Наши координаты:

E-mail: info@williamspublishing.com
WWW: http://www.williamspublishing.com
Информация для писем:
из России: 127055, Москва, ул. Лесная, д. 43, стр. 1
из Украины: 03150, Киев, а/я 152

Введение

Среди всех элементов системы Linux самым важным, пожалуй, является командная строка. Если вы организуете работу сервера под управлением Linux, то интерфейсные средства, вероятнее всего, будут исчерпываться оболочкой. Если система Linux установлена на вашей рабочей станции, то терминал, скорее всего, будет включен постоянно. Начинающим пользователям кажется, что они никогда не прибегнут к помощи командной строки. Однако чем больший опыт они приобретают, тем чаще обращаются к оболочке.

Оболочка во многом определяет богатые возможности и гибкость системы Linux. С помощью командной строки можно выполнять действия, которые были бы немыслимы при работе с графическим пользовательским интерфейсом. Независимо от того, насколько мощны KDE или GNOME (а также IceWM, или XFCE, или одна из множества других оконных сред), оказывается, что многие действия гораздо быстрее и эффективнее выполнить, пользуясь только командной строкой. Если вы хотите освоить Linux, то начинать изучение надо со средств оболочки.

Традиционный метод получения информации о командах — вызов страниц справочного руководства. Хотя информация, представленная на них, очень полезна, зачастую ее не хватает. Причина — в отсутствии примеров. Конечно, небольшие примеры кое-где встречаются, но для фрагментов кода значительного объема в справочном руководстве

не нашлось места. В связи с этим у пользователей любого уровня квалификации возникает проблема: к одному источнику надо обратиться, чтобы найти список опций и их описание, а к другому — чтобы увидеть, как эти опции используются в реальных задачах.

В этой книге приведены примеры, которых так недостает справочному руководству. Я использую Linux около десяти лет и понял, что для меня наиболее приемлема следующая конфигурация операционной системы. На верхнем уровне я, приверженец командной строки, поместил KDE, настроив его так, чтобы при регистрации в системе автоматически запускался консольный терминал. Я часто слышу жалобы пользователей Linux на нехватку примеров в справочном руководстве. Многие предлагали мне написать книгу, в которой самые важные команды Linux иллюстрировались бы многочисленными примерами. На их просьбы я отвечал коротко: "Уже пишу".

Результат моих усилий вы держите в руках. Эта книга о командах Linux, которые необходимо знать. Использование каждой из них поясняется на примерах. Данная книга — всего лишь справочник, который пригодится вам сейчас и в ближайшие годы, и все же я надеюсь, что вы получите некоторое удовольствие, читая ее.

На кого рассчитана книга

Я старался написать книгу так, чтобы она была полезна как для новичков, только приступающих к изучению Linux, так и для опытных пользователей, применяющих оболочку для решения разных задач: от администрирования до вызова игр. Если вы только начали изучать Linux, эта книга расскажет вам о возможностях оболочки; если же вы работаете с этой системой многие годы, она напомнит вам о средствах, о которых вы давно забыли, или научит некоторым специальным приемам работы.

В настоящее время существуют различные оболочки: csh, tcsh, zsh и многие другие. Я использую оболочку bash (Bourne Again Shell), по умолчанию включаемую практически в каждый дистрибутивный пакет Linux. Эта оболочка хороша не только тем, что используется повсеместно: она предоставляет богатые возможности и обеспечивает гибкость в работе. Научившись работать с bash, вы без труда перейдете к любой другой оболочке, но саму ее в любом случае знать необходимо.

При написании данной книги я использовал K/Ubuntu, но рассматриваемые в ней команды обязательно будут работать и в вашей версии системы. Единственное различие связано с работой от имени пользователя root. Вместо регистрации под именем root K/Ubuntu предполагает использование команды sudo. Другими словами, вместо того, чтобы вызывать `lsof firefox` от имени root, пользователь K/Ubuntu задает команду `sudo lsof firefox`.

Учитывая, что большинство пользователей применяют версии, отличные от K/Ubuntu, я привожу команды так, как будто их вызывает пользователь root, т.е. не указываю ключевое слово sudo. Если перед командой стоит символ #, это означает, что в системе зарегистрирован пользователь root; именно под этим именем вам придется войти в систему, чтобы выполнить данную команду. При работе с K/Ubuntu и другими подобными пакетами в этом случае указывается ключевое слово sudo.

Основные соглашения

В данной книге использовались следующие соглашения.

- Для того чтобы выделить программный код на фоне обычного текста, используется моноширинный шрифт. Символы, представленные в книге таким шрифтом, отображаются на экране компьютера. Например, они присутствуют в следующих фрагментах:

- По умолчанию команда `df` выводит результаты в килобайтах, однако они будут проще для восприятия, если использовать опцию `-h` (или `--human-readable`).
- `Этот текст выводится на экран терминала.`
- Кроме того, в книгу включены фрагменты дополнительной информации, имеющей отношение к основному материалу.

На заметку

Представлены интересные сведения, связанные с текущим текстом.

Совет

Описывается способ, позволяющий проще или быстрее решить конкретную задачу.

Внимание!

Предупреждение о возможных проблемах. Зная о них, вы сможете быстрее устранить ошибки.

1

Общие сведения о работе с командной строкой

Перед тем как вплотную приступить к изучению оболочки bash, необходимо обсудить некоторые особенности операционной системы. Зная их, вы сможете более эффективно работать с материалом этой книги. Некоторые из этих особенностей вам известны, другие покажутся не столь очевидными.

Файлы и ничего кроме файлов

Все, с чем вы встретитесь в системе Linux, — это файлы. Абсолютно все! Очевидно, что текстовый документ — это файл. Изображения, аудиоданные в формате MP3 и видеофрагменты — это несомненно файлы.

Но как насчет каталогов? Оказывается, это тоже файлы — файлы специальных типов, содержащие информацию о других файлах. Дисковые устройства — это большие файлы. Сетевые соединения — тоже файлы. Даже исполняемый процесс — это файл.

С точки зрения системы Linux файл представляет собой поток битов или байтов. Система не интересуется тем, что означает каждый байт, — это забота конкретных программ, выполняющихся в операционной системе. Для Linux

и документ, и сетевое соединение — всего лишь файлы. Как обрабатывать текстовый документ, знает редактор, а сетевое приложение умеет работать с сетевым соединением.

На протяжении всей книги будут упоминаться файлы. Подобные упоминания можно воспринимать так: "файлы, каталоги, подкаталоги и любые другие объекты". Многие из команд, которые мы вскоре рассмотрим, одинаково хорошо работают и с документами, и с каталогами, поэтому смело экспериментируйте с ними.

Максимальная длина имени файла

Те, кто имеет за плечами опыт работы с системой MS-DOS, помнят, что в ней длина имени файла была ограничена восемью символами, за которыми могли следовать три символа расширения. В результате получались "чрезвычайно информативные" имена, например `MSRSUME1.DOC`. В системах Mac, предшествующих OS X, имя файла могло содержать до 31 символа. Несмотря на то что имена файлов были достаточно длинными, иногда приходилось прибегать к сокращениям, расшифровку которых вскоре уже никто не помнил.

В системе Linux (и, конечно же, Unix) имена файлов могут насчитывать до 255 символов. Этого с лихвой хватает, чтобы сформировать имя, полностью описывающее назначение файла. Если вы используете хотя бы половину допустимой длины, у вас возникнет другая проблема — как разместить имя на экране дисплея. Если же этот вопрос вас не волнует, смело формируйте любое имя и следите лишь за тем, чтобы вам было удобно с ним работать.

На практике желательно, чтобы имена файлов не превышали 80 символов — именно столько их помещается в строке терминала. Если вы уложитесь в 80 символов, часть имени файла будет перенесена на другую строку. Ограничивать

длину имени — не требование, а всего лишь совет. Все 255 символов в вашем полном распоряжении.

Регистр символов в именах файлов

В отличие от Windows и Mac OS в системе Linux имена файлов чувствительны к регистру символов. В частности, вы можете встретить в одном каталоге все три приведенных ниже файла.

- `bookstobuy.txt`
- `BooksToBuy.txt`
- `BoOkStObUy.txt`

С точки зрения файловой системы Linux это различные имена. Если вы попытаетесь создать файлы с этими же именами в Windows или Mac OS, то для файла `BooksToBuy.txt` система предложит задать другое имя или отказаться от попыток создать его. Причина в том, что в каталоге на этот момент находится файл `bookstobuy.txt`.

Чувствительность к регистру символов также означает, что при вводе команд они должны в точности совпадать с именами файлов, поддерживающих их. Так, например, удаляя файл с помощью команды `rm`, нельзя вводить `RM`, `Rm` или `rM`. Надо также следить за написанием имен, задаваемых в качестве параметров. Если вы захотите удалить файл `bookstobuy.txt`, а укажете имя `BooksToBuy.txt`, вы лишитесь совсем не того файла, с которым предполагали расстаться.

Из сказанного можно сделать вывод, что Linux требует точности. Кстати, точность — это совсем не плохо. В то же время Linux обеспечивает такую степень гибкости, которую нельзя получить при работе с другими системами. Подобное сочетание необходимой точности и предоставляемой гибкости делает Linux очень удобным в применении,

однако некоторые пользователи поначалу испытывают неудобства при работе с этой системой.

Специальные символы в именах файлов

В каждой операционной системе определены символы, которые нежелательно или вовсе нельзя использовать в именах файлов и каталогов. Так, в Mac OS к числу подобных запрещенных символов относится двоеточие (:), а пользователям Windows нельзя включать в имена файлов обратную косую черту (\). Есть такие символы и в системе Linux. Перед тем как переходить к их рассмотрению, уточним сначала, какие знаки наверняка могут присутствовать в именах файлов. Это числа, буквы (как верхнего, так и нижнего регистра), точка и знак подчеркивания. Что же касается остальных символов, в них нельзя быть полностью уверенным. Возможно, некоторые из них не станут источником проблем, с другими возникнут сложности из-за того, что та или иная оболочка будет интерпретировать их специальным образом, а некоторые символы недопустимо применять ни при каких обстоятельствах.

Безусловно запрещена косая черта, так как этот символ используется для разделения каталогов и файлов. Предположим, что вы создали список книг, которые хотите приобрести, и собираетесь сохранить этот список в виде файла с `books/to_buy.txt`. Такое имя вы выбрали потому, что собираетесь впоследствии создать также файлы `books/on_loan.txt` и `books/lost.txt`. Однако попытка создать файл `/home/scott/documents/books/to_buy.txt` окончится неудачей. Оболочка посчитает, что `books` — это каталог в составе каталога `documents`, и не сможет найти его.

Вместо косой черты следует ввести знак подчеркивания (этот символ без проблем был интерпретирован в час-

ти имени файла to_buy). Можно также и вовсе обойтись без разделителей и назвать файл booksToBuy.txt или BooksToBuy.txt.

Для формирования имени файла можно использовать дефисы, например books-to-buy.txt, но знаки подчеркивания предпочтительнее. Если вы все же хотите использовать именно символ -, не помещайте его в начале имени файла (-books_to_buy.txt) или после пробела (books - to buy). Как вы увидите в дальнейшем, знак - служит признаком опции при вводе команды. Так, если вы собираетесь удалить файл и зададите команду rm -books_to_buy.txt, оболочка отобразит следующее сообщение об ошибке:

```
rm: invalid option -- b
```

Если необходимо, вы можете включать в имена файлов пробелы, например, допустимо создать файл books to buy.txt, однако при этом на вас возлагается дополнительная обязанность — сообщать оболочке, что пробелы являются частью имени. Дело в том, что оболочка обычно интерпретирует пробелы как разделители между параметрами. При попытке удалить файл books to buy.txt, оболочка интерпретирует соответствующую команду следующим образом: удалить файл books, затем файл to и, наконец, файл buy.txt. Мало того, что файл books to buy.txt останется на диске, вы можете случайно удалить нужные вам файлы.

Как же поступать с пробелами в именах файлов? Этот же вопрос возникает по отношению к символам * и ?, применение которых будет рассмотрено в следующем разделе. Одинарная и двойная кавычки также имеют специальное назначение. Поступать со всеми указанными выше символами можно по-разному, но лучше всего не использовать их в именах файлов. Если по каким-то причинам обойтись без них нельзя, надо указывать перед каждым символом

специального назначения обратную косую черту. Этим вы сообщите оболочке о том, что специальное значение символа надо игнорировать и интерпретировать его как обычный знак. Вводить команды подобным образом довольно утомительно, так как надо убедиться, что обратная косая черта указана везде, где необходимо.

```
$ rm Why\ shouldn\'t\ I\ name\ files\ with\ \*\?.txt
```

Есть и более простой способ — поместить имя файла в кавычки. Результат будет таким же, как если бы вы предварили каждый символ специального назначения обратной косой чертой.

```
$ rm "Why shouldn't I name files with *?.txt"
```

Такой подход даст нужный результат, нельзя лишь при вводе команды забывать о кавычках. Несмотря на обилие решений, самое лучшее из них — вовсе не включать символы специального назначения в имя файла. В табл 1.1 перечислены некоторые из таких символов и приведены рекомендации по работе с ними.

Таблица 1.1. Использование специальных символов в именах файлов

Символ	Рекомендации по использованию
/	Нельзя использовать ни при каких обстоятельствах
\	Должен быть предварен таким же символом. Применять не рекомендуется
-	Нельзя использовать в начале имени файла или каталога
[]	Каждый из этих символов должен быть предварен обратной косой чертой. Применять не рекомендуется

Окончание табл. 1.1

Символ	Рекомендации по использованию
{}	Каждый из этих символов должен быть предварен обратной косой чертой. Применять не рекомендуется
*	Должен быть предварен обратной косой чертой. Применять не рекомендуется
?	Должен быть предварен обратной косой чертой. Применять не рекомендуется
'	Должен быть предварен обратной косой чертой. Применять не рекомендуется
"	Должен быть предварен обратной косой чертой. Применять не рекомендуется

Символы групповых операций

Предположим, что в одном из каталогов на вашем компьютере содержатся двенадцать файлов с изображениями и один текстовый файл.

```
libby1.jpg
libby2.jpg
libby3.jpg
libby4.jpg
libby5.jpg
libby6.jpg
libby7.jpg
libby8.jpg
libby9.jpg
libby10.jpg
libby11.jpg
libby12.jpg
libby1.txt
```

Представьте себе, что вам надо удалить эти файлы, используя команду rm (она будет рассмотрена в следующей главе). Конечно, файлы можно удалять по одному, но это утомительно, да и компьютер как раз и предназначен для автоматизации рутинных операций. В данном случае целесообразно применить символы групповых операций, которые позволяют задать с помощью одного параметра команды одновременно несколько файлов.

Групповые операции задаются посредством звездочки (*), знака вопроса (?) и квадратных скобок ([ ]). Рассмотрим подробнее эти символы.

Звездочка отмечает любое (в том числе нулевое) количество любых символов. В табл. 1.2 приведено несколько примеров использования звездочки и показано, на какие файлы воздействует соответствующая команда.

Таблица 1.2. Использование символа * в составе команды

Команда	Файлы
`rm libby1*.*`	`libby10.jpg` — `libby12.jpg`, а также `libby1.txt`
`rm libby*.jpg`	`libby1.jpg` — `libby12.jpg`, но не `libby1.txt`
`rm *txt`	`libby1.txt`, но не `libby1.jpg` — `libby12.jpg`
`rm libby*`	`libby1.jpg` — `libby12.jpg`, а также `libby1.txt`
`rm *`	Все файлы в каталоге

Символ ? соответствует одному произвольному символу. Примеры его применения приведены в табл. 1.3.

Таблица 1.3. Использование символа * в составе команды

Команда	Файлы
`rm libby1?.jpg`	`libby10.jpg` — `libby12.jpg`, но не `libby1.txt`
`rm libby?.jpg`	`libby1.jpg` — `libby9.jpg`, но не `libby10.jpg`
`rm libby?.*`	`libby1.jpg` — `libby9.jpg`, а также `libby1.txt`

Квадратные скобки позволяют задавать один символ из набора (например, [12]) или символ, принадлежащий определенному диапазону (например, [1-3]). В последнем случае границы диапазона разделяются дефисом. В табл. 1.4 приведено несколько примеров использования квадратных скобок.

Таблица 1.4. Использование квадратных скобок в составе команды

Команда	Файлы
`rm libby1[12].jpg`	`libby11.jpg` и `libby12.jpg`, но не `libby10.jpg`
`rm libby1[0-2].jpg`	`libby10.jpg` — `libby12.jpg`, но не `libby1.jpg`
`rm libby[6-8].jpg`	`libby6.jpg` — `libby8.jpg`

Эти символы будут встречаться вам на протяжении всей книги, поэтому постарайтесь запомнить их назначение, так как они существенно упрощают работу с файлами.

Выводы

В данной главе мы рассмотрели вопросы, которые имеют отношение к работе системы Linux и которые, возможно, неочевидны для некоторых начинающих пользователей. Приведенные здесь сведения, возможно, несколько прояснят материал последующих глав. Они помогут ответить на вопрос, почему система отказывается скопировать каталог, в имени которого содержатся пробелы, или как удалить одновременно 1000 файлов, или почему не работает команда `RM bookstobuy.txt`. Вооружившись минимальными знаниями, вы сможете избежать многих ошибок, которые часто допускают новички.

Теперь самое время перейти к рассмотрению конкретных команд.

2

Основные команды

Данная глава посвящена основным командам — тем, которые каждый специалист применяет по нескольку раз в день. Эти команды можно сравнить с молотком, отверткой и плоскогубцами, которые рабочий использует очень часто и старается держать под рукой. Изучив команды, рассмотренные в данной главе, вы сможете управлять оболочкой и получите массу полезных сведений о своих файлах, каталогах, а также о системном окружении.

Вывод списка файлов и каталогов

```
ls
```

Команда `ls` — одна из самых распространенных. Перед тем как выполнять какие-либо действия с файлами, вам надо выяснить, есть ли они в каталоге. Сделать это позволяет команда `ls`, которая отображает список файлов и подкаталогов, находящихся в конкретном каталоге.

На заметку

Команда `ls` может показаться очень простой, однако это впечатление обманчиво. Существует бесчисленное число вариантов этой команды, и с некоторыми из них вы вскоре познакомитесь.

Введя `ls`, вы получите сведения о содержимом текущего каталога. Сразу после регистрации в системе текущим становится ваш рабочий каталог. Задайте команду `ls`, и вы увидите информацию, подобную приведенной ниже.

```
$ ls
alias Desktop   iso       pictures program_files
todo  bin       documents music    podcasts src
videos
```

Вывод содержимого произвольного каталога

```
ls music
```

Для того чтобы узнать содержимое каталога, не обязательно, чтобы этот каталог был текущим. Предположим, что вы находитесь в своем рабочем каталоге, а хотите узнать, что содержится в каталоге `music`. Для этого достаточно ввести команду `ls` и указать после нее имя каталога, содержимое которого вы хотите определить.

```
$ ls music
Buddy_Holly   Clash   Donald_Fagen   new
```

В этом примере мы использовали относительный путь, но с тем же успехом можно указать полный, или абсолютный, путь.

```
$ ls /home/scott/music
Buddy_Holly   Clash   Donald_Fagen   new
```

Возможность указания относительного или абсолютного пути очень удобна в тех случаях, когда вам надо узнать, какие файлы содержатся в каталоге, но вы не хотите перемещаться по файловой системе. Вы не уверены, есть ли у вас видеофайл Tiger Woods? Выполните следующую

команду (~ — это псевдоним, обозначающий рабочий каталог текущего пользователя):

```
$ ls ~/videos
Ubuntu_Talk.mpeg       nerdtv_1_andy_hertzfeld
airhorn_surprise.wmv   nerdtv_2_max_levchin
apple_navigator.mov    nerdtv_3_bill_joy
b-ball-e-mail.mov      RPG_Nerds.mpeg
carwreck.mpg           tiger_woods_just_did_it.wmv
```

Файл `tiger_woods_just_did_it.wmv` — это именно то, что вам надо.

Использование символов групповых операций при определении содержимого каталога

```
ls ~/videos/*.wmv
```

Вы уже умеете найти файл в каталоге, теперь рассмотрим более быстрый способ решения той же задачи. Если вы знаете, что нужный вам файл Tiger Woods содержит данные в формате Windows Media и, следовательно, имеет суффикс .wmv, вы можете использовать символы групповых операций, чтобы вывести только файлы, оканчивающиеся на .wmv.

```
$ ls ~/videos
Ubuntu_Talk.mpeg       nerdtv_1_andy_hertzfeld
airhorn_surprise.wmv   nerdtv_2_max_levchin
apple_navigator.mov    nerdtv_3_bill_joy
b-ball-e-mail.mov      RPG_Nerds.mpeg
carwreck.mpg           tiger_woods_just_did_it.wmv
$ ls ~/videos/*.wmv
airhorn_surprise.wmv tiger_woods_just_did_it.wmv
```

Существует еще один метод, который также предполагает применение символов групповых операций: в данном случае можно искать файл, в имени которого содержится слово tiger.

```
$ ls ~/videos/*tiger*
tiger_woods_just_did_it.wmv
```

Просмотр содержимого подкаталогов

```
ls -R
```

При необходимости вы можете с помощью одной команды просмотреть содержимое нескольких подкаталогов. Предположим, что кому-то из ваших знакомых срочно потребовался ISO-файл Kubuntu. Вы вспоминаете, что вроде бы скопировали его несколько дней назад, но, чтобы быть уверенным, выполняете следующую команду (вместо `ls -R` можно задать `ls --recursive`):

```
$ ls -R ~/iso
iso:
debian-31r0a-i386-netinst.iso  knoppix  ubuntu

iso/knoppix:
KNOPPIX_V4.0.2CD.iso  KNOPPIX_V4.0.2DVD.iso

iso/ubuntu:
kubuntu-5.10-install.iso  ubuntu-5.10-install.iso
kubuntu-5.10-live.iso     ubuntu-5.10-live.iso
```

И действительно, в каталоге `~/iso/ubuntu` есть файл `kubuntu-5.10-install-i386.iso`. Опция `-R` задает рекурсивный просмотр каталога `iso`, и вы получаете сведения о содержимом этого каталога и всех его подкаталогов. Для каждого каталога отображается его путь относительно

того каталога, с которого вы начали работу, затем выводится двоеточие, а после него — содержимое каталога. Учтите, что при большом количестве подкаталогов опция рекурсивного выполнения становится бесполезной, так как на экран выводится большой объем информации и найти сведения о нужном файле проблематично. Конечно, если вам надо всего лишь убедиться, что в подкаталогах данного каталога содержится большое количество файлов, эта опция подходит как нельзя лучше, однако необходимость в подобных сведениях возникает нечасто.

Вывод содержимого каталога в один столбец

```
ls -1
```

До сих пор мы имели дело с установками команды `ls` по умолчанию, согласно которым содержимое каталога выводится в алфавитном порядке в несколько столбцов, причем между соседними столбцами помещается как минимум два пробела. Но что делать, если вам надо вывести данные в другом формате?

Если вывод в несколько столбцов вас не устраивает, вы можете отобразить результаты выполнения команды `ls` в один столбец. Для этого используется команда `ls -1` (или `ls --format=single-column`).

```
$ ls -1 ~/
bin
Desktop
documents
iso
music
pictures
src
videos
```

Если в каталоге содержится очень много файлов, да к тому же вы задали опцию рекурсивного выполнения, выводимый список может стать неуправляемым. Поэтому, убедившись, что процесс вывода данных, полученных по команде `ls -lR ~/`, грозит затянуться надолго, нажмите комбинацию клавиш <Ctrl+C>, чтобы прервать работу.

Вывод содержимого каталога с запятыми в качестве разделителей

```
ls -m
```

Еще один вариант рассматриваемой нами команды предназначен для тех, кто по каким-то причинам не хочет, чтобы выходные данные были оформлены в виде одного или нескольких столбцов. Отказаться от такого форматирования позволяет опция `-m` (или `--format=commas`).

```
$ ls -m ~/
bin, Desktop, docs, iso, music, pix, src, videos
```

Чтобы лучше запомнить данную опцию, надо принять во внимание, что буква `m` дважды встречается в слове `comma`. Эта опция пригодится в тех случаях, когда вы пишете сценарий и вам необходимо иметь имена файлов, разделенных запятыми.

Отображение скрытых файлов и каталогов

```
ls -a
```

До сих пор мы получали информацию о видимых файлах и каталогах, но необходимо помнить, что помимо них могут существовать и скрытые файлы. Подобных файлов достаточно много и в вашем рабочем каталоге. Невидимым файл становится в том случае, если его имя начинается с

точки. Если вы хотите, чтобы информация о невидимых файлах также отображалась на экране, вам следует использовать опцию -a (или --all).

```
$ ls -a ~/
.               .gimp-2.2        .openoffice.org1.9.95
..              .gksu.lock       .openoffice.org1.9
.3ddesktop      .glade2          .openoffice.org2
.abbrev_defs    .gnome           .opera
.adobe          .gnome2_private  pictures
```

Просматривая результаты, полученные с помощью команды `ls -a`, легко заметить следующее. Во-первых, эта команда отображает как скрытые, так и обычные файлы; например, в данном примере вы видите имя `.gnome` и `pictures`. Во-вторых, в списке есть элементы `.` и `..`; одна точка соответствует текущему каталогу, а две точки — каталогу, расположенному выше по иерархии, т.е. родительскому по отношению к текущему. Эти два скрытых каталога присутствуют в любом каталоге вашей системы, и удалить их невозможно. Отобразить их позволяет опция `-a`. В некоторых каталогах число скрытых файлов очень велико, а о наличии некоторых из них вы даже не догадываетесь.

Отображение информации о типах файлов

```
ls -F
```

Обычная команда `ls` не сообщает о файлах, находящихся в каталоге, ничего, кроме их имен. Отображаемая с ее помощью информация не дает даже возможности отличить обычный файл от каталога. Для того чтобы получить дополнительные сведения, надо использовать опцию `-F` (или `--classify`).

```
$ ls -F ~/bin
adblock_filters.txt  fixm3u*      pix2tn.pl*
addext*              flash.xml*   pop_login*
address_book.csv     getip*       procmail/
address_book.sxc     homesize*
programs_kill_artsd*
address_book.xls     html2text.py*
programs_usual*
```

Объем этих сведений невелик. Звездочка после имени файла говорит о том, что файл является исполняемым, а косая черта — это признак каталога. Если после имени файла не указан никакой символ, значит, это обычный файл. Символы, которые могут присутствовать в списке после имен файлов, перечислены в табл. 2.1.

Таблица 2.1. Символы, обозначающие типы файлов

Символ	Тип файла
*	Исполняемый файл
/	Каталог
@	Символьная ссылка
|	FIFO
=	Сокет

Отображение информации в цвете

```
ls --color
```

Опция -F, рассмотренная ранее, позволяла получить информацию о файле с помощью предопределенных символов. Кроме того, вы можете указать оболочке выводить информацию в цвете. Это дает дополнительные возможности для

классификации содержимого каталога. В некоторых версиях Linux вывод в цвете предусмотрен по умолчанию, но если такая установка в вашей системе отсутствует, вы можете воспользоваться опцией `--color`.

```
$ ls --color
adblock_filters.txt  fixm3u     pix2tn.pl
addext               flash.xml  pop_login
address_book.csv     getip      procmail
```

В нашем случае исполняемые файлы отображаются зеленым цветом, каталоги — синим, а обычные файлы — черным цветом (который в большинстве оболочек установлен по умолчанию). В табл. 2.2 приведены соглашения по использованию цвета для отображения типов файлов.

Таблица 2.2. Соответствие цвета типу файла

Цвет	Тип файла
Цвет, используемый оболочкой по умолчанию	Обычный файл
Green	Исполняемый файл
Blue	Каталог
Magenta	Символьная ссылка
Yellow	FIFO
Magenta	Сокет
Red	Архив (.tar, .zip, .deb, .rpm)
Magenta	Изображение (.jpg, .gif, .png, .tiff)
Magenta	Аудиофайл (.mp3, .ogg, .wav)

Совет

Если вы хотите выяснить, какому типу файла соответствует тот или иной цвет, введите команду `dircolors --print-database` и ознакомьтесь с результатами. Команду `dircolors` можно также использовать для того, чтобы изменить соответствие цветов типу файлов.

Используя опции `--color` и `-F`, вы можете быстро составить представление о том, какие типы файлов находятся в каталоге.

```
$ ls -F --color
adblock_filters.txt  fixm3u*     pix2tn.pl*
addext*              flash.xml*  pop_login*
address_book.csv     getip*      procmail/
```

Информация о правах доступа и владельцах файлов

```
ls -l
```

Вы уже знаете, как получить дополнительные сведения о содержимом каталога, но в некоторых случаях необходима более подробная информация. Сейчас мы рассмотрим, как определять размер, владельца и права доступа к файлу для различных категорий пользователей. Получить эти сведения позволяет опция `-l` (или `--format=long`).

```
$ ls -l ~/bin
total 2951
-rw-r--r--  1  scott scott   15058  2005-10-03
  18:49  adblock_filters.txt
-rwxr-xr--  1  scott root       33  2005-04-19
  09:45  addext
-rwxr--r--  1  scott scott     245  2005-10-15
  22:38  backup
```

```
drwxr-xr-x  9  scott scott    1080  2005-09-22
  14:42  bin_on_bacon
-rw-r--r--  1  scott scott  237641  2005-10-14
  13:50  calendar.ics
-rwxr-xr--  1  scott root      190  2005-04-19
  09:45  convertsize
drwxr-xr-x  2  scott scott      48  2005-04-19
  09:45  credentials
```

Опция `-l` означает "long". Как вы видите, она задает отображение подробных сведений о файлах, содержащихся в каталоге. Рассмотрим типичную строку, начиная с самого правого элемента.

Как нетрудно догадаться, заканчивается строка именем файла. Тип этого файла можно отобразить, задавая опцию `-F`, например `ls -lF`. Также можно указать и отображение в цвете (`ls -lF --color`).

Перемещаясь по строке влево, вы встретите информацию о дате и времени. Они определяют момент последней модификации файла (дата отображается в формате год-месяц-день, а время отсчитывается посредством 24-часового цикла).

Левее даты располагается число, которое обозначает размер файла в байтах. Размер приводится и для каталогов, что может вызвать определенные затруднения у начинающих пользователей. В приведенном примере размер каталога `bin_on_bacon` составляет 1080 байтов, т.е. немногим более одного килобайта, но содержится в нем 887 Кбайт данных. С другой стороны, согласно данным, полученным с помощью команды `ls -l`, размер каталога `credentials` составляет 48 байтов, но этот каталог пуст. В чем же дело?

В главе 1 было сказано, что каталоги — это специальный тип файлов, в которых находится информация о содержимом. В данном случае "содержимое" каталога

credentials исчерпывается родительским каталогом с именем .., поэтому он составляет 48 байтов, а каталог bin_on_bacon содержит более 30 файлов, поэтому его размер равен 1080 байтам.

Двигаясь влево по строке, мы встретим столбцы, в которых отображается информация о владельце файла и группе. Как видно из предыдущего примера, почти все файлы принадлежат пользователю scott и группе scott. Исключение составляют файлы addext и convertsize, для которых указаны пользователь scott и группа root.

На заметку

Принадлежность файла пользователю и группе можно изменить. Как это сделать, вы узнаете в главе 7 (для этой цели используются команды chown и chgrp).

Во втором слева столбце содержится число. Оно сообщает вам, сколько существует "жестких" ссылок на этот файл. Для каталога данное число обозначает количество файлов, содержащихся в нем.

Совет

Дополнительную информацию о "жестких" и "мягких" ссылках вы найдете в документе http://www.granneman.com/techinfo/linux/thelinuxenvironment/softandhardlinks.htm. Можно также выполнить средствами Google поиск по ключевым словам linux hard links.

Итак, мы достигли самого левого столбца. В нем отображаются права доступа к каждому файлу и каталогу. Непосвященному данная последовательность символов не скажет ничего, но она чрезвычайно информативна для тех, кто обладает необходимым минимумом знаний. В данном столбце содержится десять элементов, которые можно ус-

ловно разделить на четыре группы. Первой группе принадлежит первый символ, знаки в позициях 2–4 составляют вторую группу, символы, занимающие позиции 5–7, принадлежат третьей группе и, наконец, четвертой группе принадлежат символы в позициях 8–10. Так, например, последовательность символов для каталога `credentials` можно разделить следующим образом: `d|rwx|r-x|r-x`.

Первая группа сообщает вам тип файла. Как вы уже знаете, опции `-F` и `--color` отображают типы файлов различными способами. Еще один способ реализует опция `-l`. Символ `d` в первой позиции означает, что в данной строке представлен каталог. Если же в этой позиции находится символ `-`, значит, речь идет об обычном файле. (Даже если файл является исполняемым, команда `ls -l` отобразит в первой позиции символ `-`; в этом случае опции `-F` и `--color` предоставят более подробную информацию.) В табл. 2.3 перечислены символы, которые могут присутствовать в первой позиции строки, отображаемой по команде `ls -l`.

Таблица 2.3. Символы, представляющие права доступа, и типы файлов

Символ	Тип файла
-	Обычный файл
-	Исполняемый файл
d	Каталог
l	Символьная ссылка
s	Сокет
b	Блочное устройство
c	Символьное устройство
p	Именованный канал

Совет

Для того чтобы увидеть все или почти все буквы, приведенные в табл. 2.1, выполните команду `ls -l /dev`.

Остальные девять символов, составляющие вторую, третью и четвертую группы, задают права для владельца файла, группы и всех остальных пользователей системы. Для файла `addext`, присутствующего в рассмотренном выше примере, права заданы так: `rwxr-xr--`. Это означает, что владелец `scott` имеет права `rwx`, группа (в данном случае она тоже называется `scott`) — права `r-x`, а все остальные пользователи имеют права `r--`. Как же расшифровать эту запись?

В каждом случае буква `r` обозначает "чтение разрешено", буква `w` — "запись разрешена", а буква `x` — "выполнение разрешено". Символ `-` в соответствующей позиции означает "данное действие запрещено". Если дефис указан вместо буквы `r`, это значит "чтение запрещено". Также запрещается запись или выполнение, если символ `-` находится на месте `w` или `x`.

Вернемся еще раз к файлу `addext` и правам доступа `rwxr-xr--`. Теперь нам понятно, что владелец (`scott`) может читать, записывать или выполнять файл, члены группы (`root`) могут читать файл и запускать его на выполнение, но не записывать информацию в него. Любой другой пользователь может читать файл, а запись и запуск на выполнение ему запрещены.

Теперь, когда вы знаете, как расшифровывается информация о правах, обратите внимание на то, что некоторые сочетания символов встречаются очень часто. Например, для многих файлов установлены права `rw-r--r--`; это означает, что владелец может выполнять операции чтения и записи, а остальные пользователи, в том числе члены группы, могут только читать файл. Для программ вы часто встрети-

те права rwxr-xr-x, т.е. каждый пользователь может читать и выполнять программу, но вносить изменения в файл может только его владелец.

С каталогами дело обстоит несколько по-другому. Права r, w и x, относящиеся к файлу, интуитивно понятны: они соответствуют чтению, записи и выполнению. Но как выполнить каталог?

Начнем с самого простого — права r. В случае каталога r означает, что пользователь может просматривать содержимое каталога с помощью команды ls. Символ w говорит о том, что пользователь может включать в каталог новые файлы, а также переименовывать и удалять существующие. Символ x означает доступ к каталогу, т.е. право выполнять команды, выполняющие определенные действия с файлами из этого каталога, или обращаться к подкаталогам данного каталога.

Как видите, опция -l предоставляет обширные возможности, но она оказывается еще более полезной в сочетании с другими опциями. Вы уже знаете опцию -a, которая отображает все файлы, содержащиеся в каталоге, поэтому для вас очевиден будет результат выполнения команды ls -la (вместо -la можно также задать --format=long --all).

```
$ la -la ~/
drwxr-xr-x   2  scott scott   200  2005-07-28
  01:31  alias
drwx------   2  root  root     72  2005-09-16
  19:14  .aptitude
-rw-r--r--   1  scott scott  1026  2005-09-25
  00:11  .audacity
drwxr-xr-x  10  scott scott   592  2005-10-18
  11:22  .Azureus
-rw-------   1  scott scott  8800  2005-10-18
  19:55  .bash_history
```

На заметку

Если владельцем файла является `scott` или если так называется группа, которой принадлежит файл, то эти данные для экономии места будут удаляться из листингов, приведенных ниже.

Вывод информации в обратном порядке

```
ls -r
```

Если вам не подходит информация о файлах, выводимая в алфавитном порядке, вы можете изменить порядок на обратный с помощью опции `-r` (или `--reverse`).

```
$ ls -lar ~/
-rw------- 8800  2005-10-18  19:55  .bash_history
drwxr-xr-x  592  2005-10-18  11:22  .Azureus
-rw-r--r-- 1026  2005-09-25  00:11  .audacity
drwx------   72  2005-09-16  19:14  .aptitude
drwxr-xr-x  200  2005-07-28  01:31  alias
```

На заметку

Обратите внимание, что данная опция задается буквой нижнего регистра (`-r`, а не `-R`). Опция `-r` задает обратный порядок вывода, а `-R` означает рекурсивное выполнение.

Когда вы используете опцию `-l`, имена файлов и каталогов располагаются по алфавиту. Опция `-r` изменяет этот порядок на обратный, но сортировка по-прежнему производится по имени файла.

Сортировка содержимого каталога по суффиксам имен файлов

```
ls -x
```

Имя файла — не единственный признак, по которому можно выполнять сортировку. Для сортировки можно применять суффикс имени. Другими словами, вы можете сообщить команде `ls` сгруппировать вместе все файлы, оканчивающиеся на `.doc`, затем файлы, оканчивающиеся на `.jpg`, и, наконец, файлы, имена которых оканчиваются символами `.txt`. Для этой цели надо использовать опцию `-X` (или `--sort=extension`); если вы захотите отобразить результаты в обратном порядке, следует также добавить опцию `-r` (или `--reverse`).

```
$ ls -lX ~/src
drwxr-xr-x      320  2005-10-06  22:35  backups
drwxr-xr-x     1336  2005-09-18  15:01  fonts
-rw-r--r--  2983001  2005-06-20  02:15
  install.tar.gz
-rw-r--r--  6683923  2005-09-24  22:41
  DuckDoom.zip
```

Данные, выведенные в результате выполнения описанной команды, имеют две интересные особенности. Во-первых, первыми в списке находятся каталоги, так как для них суффикс не указан, затем следуют файлы с суффиксами. Во-вторых, обратите внимание на файл `installdata.tar.gz`. В его имени содержатся две точки, но команда `ls` учитывает только символы `.gz`, которыми оканчивается имя файла.

Сортировка по дате и времени

```
ls -t
```

Возможность сортировки по суффиксу удобна, но в некоторых случаях желательно расположить содержимое каталога по возрастанию или по убыванию даты и времени. Для того чтобы сделать это, надо наряду с опцией `-l` указать опцию `-t` (или `--sort=time`). Для того чтобы расположить отсортированные данные в обратном порядке, надо кроме `-l` задать `-tr` (или `--sort=time --reverse`).

```
$ ls -latr ~/
-rw-------  8800  2005-10-18  19:55  .bash_history
drwx------   368  2005-10-18  23:12  .gnupg
drwxr-xr-x  2760  2005-10-18  23:14  bin
drwx------   168  2005-10-19  00:13  .Skype
```

Все файлы, за исключением последнего, были модифицированы в один и тот же день. Если бы опция `-r` не была указана, последний файл отобразился бы первым.

На заметку

Обратите внимание, что в последнем примере было указано четыре опции `-latr`. Их можно задать и по отдельности, `-l -a -t -r`, но при этом пришлось бы вводить лишние дефисы и пробелы. Гораздо быстрее и проще объединить их в одну последовательность символов. Существуют также "длинные" варианты опций. Такая опция начинается с двух дефисов и состоит из одного или двух слов. Их нельзя объединять и надо задавать раздельно. Другими словами, команду, использованную в предыдущем примере, можно задать и так: `ls -la --sort=time --reverse`.

Сортировка содержимого каталога по размеру файлов

```
ls -s
```

Содержимое каталога можно отсортировать также и по размеру файлов. Сделать это позволяет опция `-S` (или `--sort=size`).

```
$ ls -laS ~/
-rw-r--r--  109587  2005-10-19  11:53
  .xsession-errors
-rw-------   40122  2005-04-20  11:00  .nessusrc
-rwxr--r--   15465  2005-10-12  15:45  .vimrc
-rw-------    8757  2005-10-19  08:43
  .bash_history
```

При сортировке по размеру самый большой файл указывается первым. Если задан обратный порядок следования, т.е. если указана опция `-r`, вверху списка окажется файл самого меньшего размера.

Представление размеров файлов в килобайтах, мегабайтах и гигабайтах

```
ls -h
```

В списке из предыдущего раздела содержался файл `.vimrc` размером 15465 байтов, т.е. порядка 15 Кбайт. Преобразовывать в уме байты в килобайты, мегабайты и гигабайты не всегда удобно. Часто бывает предпочтительнее использовать опцию `-h` (или `--human-readable`), которая представляет информацию в виде, удобном для восприятия пользователем.

```
$ ls -lash ~/
-rw-r--r--  100K  2005-10-19  11:44
  .xsession-errors
-rw-------   40K  2005-04-20  11:00  .nessusrc
-rwxr--r--   16K  2005-10-12  15:45  .vimrc
-rw-------  8.6K  2005-10-19  08:43  .bash_history
```

В данном примере размер файлов представлен в килобайтах, на что указывает буква K после соответствующего числа. Если бы файлы были достаточно большими, вы бы увидели букву M (мегабайты) и даже G (гигабайты). Возможно, вы удивитесь, как 40122 байта (размер файла `.nessusrc`) превратились в 40 Кбайт; ведь в килобайте 1024 байта, и, разделив 40122 на 1024, мы получим 39,1816406 байта. Однако команда `ls -h` округляет данное значение до 40K. Заметьте, что мегабайт содержит 1048576 байтов, а гигабайт — 1073741824 байта и при использовании этих единиц измерения также возможны подобные округления.

На заметку

В файле `~/.bashrc` я задал несколько псевдонимов, которые уже много лет упрощают мою работу в системе. Возможно, прочитав материал этого раздела, вам также захочется создать псевдонимы, подобные приведенным ниже.

```
alias l='ls -F'
alias ll='ls -lF'
alias la='ls -aF'
alias ll='ls -laFh'
alias ls='ls -F'
```

Определение пути к текущему каталогу

```
pwd
```

Может случиться так, что, перемещаясь по каталогам и выводя на экран их содержимое, вы забудете, в каком месте файловой системы вы находитесь в данный момент. Как узнать, какой из каталогов является текущим? Сделать это позволяет команда pwd, имя которой расшифровывается как print working directory (вывести текущий каталог).

На заметку

Слово "print" в названии print working directory означает вывод на экран, а не на принтер.

Команда pwd отображает полный путь к текущему каталогу. Вряд ли вы будете использовать эту команду очень часто, но время от времени она оказывается довольно полезной.

```
$ pwd
/home/scott/music/new
```

Переход к другому каталогу

```
cd
```

Отобразить содержимое любого каталога можно, указав путь к нему в качестве параметра команды `ls`. Однако часто возникает необходимость перейти в другой каталог, т.е. сделать его текущим. В этих случаях надо применять команду `cd` — одну из самых популярных команд любой оболочки.

Команда `cd` очень проста в использовании, в качестве параметра ей передается каталог, который необходимо сде-

лать текущим. при этом можно использовать как относительный (например, `cd src` или `cd ../../`), так и абсолютный путь (например, `cd /tmp` или `cd /home/scott/bin`).

Переход в рабочий каталог

```
cd ~
```

Некоторые полезные возможности команды `cd` могут несколько упростить работу в системе. Независимо от того, какой каталог является текущим, выполнив команду `cd`, вы немедленно перейдете в свой рабочий каталог. Эта возможность позволяет сэкономить время. Можно также использовать команду `cd ~`, так как ~ является сокращением, обозначающим "мой рабочий каталог".

```
$ pwd
/home/scott/music
$ cd ~
$ pwd
/home/scott
```

Переход к предыдущему каталогу

```
cd -
```

Еще один полезный вариант рассматриваемой здесь команды — это `cd -`. Она осуществляет переход к предыдущему каталогу, после чего автоматически вызывает команду `pwd`, которая выводит информацию о новом каталоге (правда, в данном случае его уместнее назвать не новым, а старым). Действие команды `cd -` демонстрирует следующий пример:

```
$ pwd
/home/scott
$ cd music/new
$ pwd
/home/scott/music/new
$ cd -
/home/scott
```

Команда `cd -` удобна тогда, когда вам надо перейти в другой каталог, выполнить в нем некоторые действия, а затем вернуться в тот каталог, с которым вы работали ранее. Информация о каталоге, выводимая на экран, позволяет вам удостовериться в правильности выполненных действий.

Изменение сведений о времени

`touch`

Команда `touch` не относится к тем, которые используются ежедневно, но мы будем применять ее в этой книге, поэтому удобнее всего ознакомиться с ней именно сейчас. Основное назначение данной команды — установка времени доступа и модификации файла, однако мы будем применять ее с другой целью. Как ни странно, вспомогательная функция команды оказывается более важной для нас, чем основная.

На заметку

Применять команду `touch` к файлу можно только в том случае, если вы имеете право записывать информацию в этот файл. В противном случае при попытке выполнить команду `touch` возникнет ошибка.

Для того чтобы одновременно изменить время доступа и время модификации файла (или каталога), надо выполнить команду `touch`, не задавая никаких опций.

```
$ ls -l ~/
drwxr-xr-x    848  2005-10-19  11:36  src
drwxr-xr-x   1664  2005-10-18  12:07  todo
drwxr-xr-x    632  2005-10-18  12:25  videos
-rw-r--r--    239  2005-09-10  23:12  wireless.log
$ touch wireless.log
$ ls -l ~/
drwxr-xr-x    848  2005-10-19  11:36  src
drwxr-xr-x   1664  2005-10-18  12:07  todo
drwxr-xr-x    632  2005-10-18  12:25  videos
-rw-r--r--    239  2005-10-19  14:00  wireless.log
```

В результате выполнения приведенной выше команды изменилось время доступа и время модификации файла `wireless.log`. Это не очевидно, так как по команде `ls -l` отображается только время модификации файла. Файл не использовался около месяца, но команда `touch` изменила информацию о дате и времени.

При необходимости можно изменить время доступа и время модификации по отдельности. Если надо изменить только время доступа, следует указать опцию `-a` (или `--time=access`), а чтобы установить только время модификации, надо использовать опцию `-m` (или `--time=modify`).

Установка произвольного времени для файла

`touch -t`

Заметьте, что команда `touch` не ограничивает вас текущими датой и временем. Вы можете выбрать любую дату, используя опцию `-t` и задавая значение в следующем формате: `[[CC]YY]MMDDhhmm[.ss]`. Назначение основных его элементов приведено в табл. 2.4.

Таблица 2.4. Элементы шаблона, используемые для установки времени доступа и времени модификации файла

Символы	Значение
CC	Первые две цифры года, задаваемого четырьмя цифрами
YY	Год, задаваемый двумя цифрами: • значение 00–68 предполагает первые две цифры — 20; • значение 69–99 предполагает первые две цифры — 19; • отсутствующее значение предполагает текущий год
MM	Месяц (01–12)
DD	День (01–31)
hh	Часы (01–23)
mm	Минуты (00–59)
ss	Секунды (00–59)

Важно помнить, что в тех случаях, когда в шаблоне предусмотрены две цифры, а значение представляется лишь одной цифрой, надо указывать ведущий нуль. В противном случае опция будет интерпретирована неправильно. Ниже представлено несколько примеров использования опции `-t`.

```
$ ls -l
-rw-r--r-- 239 2005-10-19 14:00 wireless.log
$ touch -t 197002160701 wireless.log
$ ls -l
-rw-r--r-- 239 1970-02-16 07:01 wireless.log
$ touch -t 9212310000 wireless.log
$ ls -l
-rw-r--r-- 239 1992-12-31 00:00 wireless.log
$ touch -t 3405170234 wireless.log
```

```
$ ls -l
-rw-r--r-- 239 2034-05-17 02:34 wireless.log
$ touch -t 10191703 wireless.log
$ ls -l
-rw-r--r-- 239 2005-10-19 17:03 wireless.log
```

Сначала мы выясняем текущую дату и время для файла `wireless.log` (2005-10-19 14:00). Затем возвращаемся во времени на 35 лет (1970-02-16 07:01), а после этого переходим более чем на двадцать лет вперед (1992-12-31 00:00). Как видите, мы можем совершить даже скачок в будущее (2034-05-17 02:34). Это будет время повсеместного использования компьютеров под управлением Linux, время всеобщего мира и открытого кода. В конце концов, мы возвратимся в настоящее время.

Приведенный пример демонстрирует некоторые особенности работы с командой `touch`. Когда мы возвращались назад более чем на три десятилетия, мы задавали четырьмя цифрами год (1970), кроме того, указали месяц (02), число (16), количество часов (07) и минут (01). Секунды можно не указывать, и мы не делали этого. Данная команда — единственная, в которой год определялся с помощью четырех цифр. Значение 92 в 9212310000 попадает в диапазон 69–99, поэтому в качестве базового выбирается 1900 год. Число 34 в составе значения 3405120234 принадлежит диапазону 0–68, поэтому базовым считается 2000 год. В последней команде год не задается вовсе, указывается только месяц (10), число (19), количество часов (17) и минут (03). В этом случае команда `touch` принимает текущий год. Зная, как пользоваться командой `touch`, мы можем изменять время доступа или модификации файла.

Создание нового пустого файла

```
touch
```

Необходимость изменять дату возникает достаточно редко. Однако команда touch имеет еще одно, гораздо более интересное применение. С ее помощью можно воздействовать на файл, который еще не создан. В результате команда создаст новый файл с указанным именем.

```
$ ls -l ~/
drwxr-xr-x   848  2005-10-19  11:36  src
drwxr-xr-x   632  2005-10-18  12:25  videos
$ touch test.txt
$ ls -l ~/
drwxr-xr-x   848  2005-10-19  11:36  src
-rw-r--r--     0  2005-10-19  23:41  test.txt
drwxr-xr-x   632  2005-10-18  12:25  videos
```

Зачем же нужно использовать команду touch подобным способом? Причины могут быть разными. Возможно, вы захотите сначала создать файл, а потом записать в него данные. Не исключено, что вам понадобится несколько файлов для того, чтобы проверить, как работает незнакомая вам команда. По мере того как вы будете подробнее изучать оболочку, вы придумаете новые причины для создания пустого файла.

Создание нового каталога

```
mkdir
```

Команда touch может создать пустой файл, но как насчет нового каталога? В этом вам поможет команда mkdir.

```
$ ls -l
drwxr-xr-x   848  2005-10-19  11:36  src
drwxr-xr-x   632  2005-10-18  12:25  videos
$ mkdir test
$ ls -l
drwxr-xr-x   848  2005-10-19  11:36  src
drwxr-xr-x    48  2005-10-19  23:50  test
drwxr-xr-x   632  2005-10-18  12:25  videos
```

На заметку

В большинстве систем новые каталоги, созданные посредством команды `mkdir`, предоставляют владельцу права чтения, записи и выполнения, а членам группы — только права чтения и выполнения. Если вы хотите назначить права по-другому, вам следует воспользоваться командой `chmod`, которая будет рассмотрена в главе 7.

К счастью, в данном случае оболочка следит за правильностью ваших действий: при попытке создать уже существующий каталог, команда завершится с ошибкой и выведет предупреждающее сообщение.

```
$ mkdir test
mkdir: cannot create directory 'test': File exists
```

Создание нового каталога и необходимых подкаталогов

```
mkdir -p
```

Если вы хотите создать каталог, в нем новый подкаталог, а в нем еще один подкаталог, может показаться, что эту рутинную задачу следует решать так: создать первый каталог с помощью команды `mkdir`, перейти в него, вызвав команду `cd`, создать подкаталог, опять же использовав для этого команду `mkdir`, сделать его текущим и вызвать коман-

ду mkdir для создания очередного подкаталога. К счастью, в команде mkdir предусмотрена удобная опция -p (или --parents), существенно упрощающая весь процесс.

```
$ ls -l
drwxr-xr-x   848  2005-10-19  11:36  src
$ mkdir -p pictures/personal/family
$ ls -l
drwxr-xr-x    72  2005-10-20  00:12  pictures
drwxr-xr-x   848  2005-10-19  11:36  src
$ cd pictures
$ ls -l
drwxr-xr-x    72  2005-10-20  00:12  personal
cd personal
$ ls -l
drwxr-xr-x    48  2005-10-20  00:12  family
```

Информация о действиях, выполняемых командой mkdir

```
mkdir -v
```

Можно ли еще более упростить данную задачу? Оказывается, можно, если задать опцию -v (или --verbose). Эта опция сообщит о каждом выполненном действии, так что вам не надо будет проверять результат вызова команды.

```
$ mkdir -pv pictures/personal/family
mkdir: created directory 'pictures'
mkdir: created directory 'pictures/personal'
mkdir: created directory
  'pictures/personal/family'
```

Для ленивых пользователей работать с системой Linux — одно удовольствие. Создается впечатление, что чем ленивее пользователь, тем больше идет к нему навстречу система. Подтверждение тому — приведенный выше пример.

Копирование файлов

cp

Каждому пользователю, независимо от того, с какой операционной системой он работает, то и дело приходится копировать файлы. Команда cp — одна из главных в оболочке Linux. Она предназначена для копирования файлов и каталогов. Самый простой способ скопировать файл — ввести команду cp, указав после нее имя копируемого файла, а затем имя нового файла, который должен получиться в результате копирования. Другими словами, данная команда задается следующим образом: cp *файл_для_копирования новый_файл*. Ее также можно описать как cp *файл_источник целевой_файл*.

```
$ pwd
/home/scott/libby
$ ls
libby.jpg
$ cp libby.jpg libby_bak.jpg
$ ls
libby_bak.jpg    libby.jpg
```

Данный пример чрезвычайно прост. Изображение копируется в тот же каталог, который содержит исходный файл. Можно также копировать файлы в другой каталог либо копировать их из каталога, который не является текущим.

```
$ pwd
/home/scott
$ ls ~/libby
libby_bak.jpg    libby.jpg
$ cp pix/libby_arrowrock.jpg libby/arrowrock.jpg
$ ls ~/libby
arrowrock.jpg    libby_bak.jpg    libby.jpg
```

Поскольку в данном случае файл копируется в другой каталог, его имя может оставаться неизменным. В предыдущем примере нам пришлось задать новое имя — `libby_bak.jpg`, так как исходный файл, `libby.jpg`, копировался в тот же каталог.

Если вам надо скопировать файл в текущий каталог из какой-то другой точки файловой системы, используйте в качестве имени целевого каталога точку. (Вы, конечно же, помните, что точка означает текущий каталог? Теперь эта информация пригодилась.) Очевидно, что если вы укажете в качестве второго параметра команды `cp` точку, имя целевого файла будет совпадать с именем исходного.

```
$ pwd
/home/scott/libby
$ ls
libby_bak.jpg  libby.jpg
$ cp pix/libby_arrowrock.jpg .
$ ls
arrowrock.jpg   libby_bak.jpg   libby.jpg
```

Если файл, полученный в результате копирования, должен быть помещен в конкретный каталог, вы можете не указывать имя целевого файла, достаточно задать имя каталога.

```
$ ls -l
drwxr-xr-x   224  2005-10-20  12:34  libby
drwxr-xr-x   216  2005-09-29  23:17  music
drwxr-xr-x  1.6K  2005-10-16  12:34  pix
$ ls libby
arrowrock.jpg   libby.jpg
$ cp pix/libby_on_couch.jpg libby
$ ls libby
arrowrock.jpg   libby.jpg   libby_on_couch.jpg
```

Выполняя команду, приведенную в данном примере, надо быть уверенным, что каталог `libby` существует, в противном случае файл `libby_on_couch.jpg` будет скопирован в файл `libby` и помещен в текущий каталог.

Копирование файлов с использованием символов групповых операций

```
cp *
```

Как уже говорилось выше, система Linux поощряет ленивых пользователей. В частности, она предоставляет возможность скопировать одновременно несколько файлов, используя символы групповых операций. Если вы грамотно именовали файлы, то сэкономите время при их копировании, так как сможете с помощью одной команды обрабатывать несколько файлов.

```
$ pwd
/home/scott/libby
$ ls ~/pix
arrowrock.jpg   by_pool_03.jpg   on_floor_03.jpg
by_pool_01.jpg  on_floor_01.jpg  on_floor_04.jpg
by_pool_02.jpg  on_floor_02.jpg
$ ls
arrowrock.jpg   libby.jpg   libby_on_couch.jpg
$ cp ~/pix/by_pool*.jpg
$ ls
arrowrock.jpg   by_pool_02.jpg   on_couch.jpg
by_pool_01.jpg  by_pool_03.jpg   libby.jpg
```

Символ * — не единственный из тех, которые можно применять для групповых операций. Вы можете задавать имена файлов, используя квадратные скобки. Например, если вы хотите скопировать изображения `libby_on_floor`

с номерами 01, 02, 03, но не 04, вы можете сделать это так, как показано в следующем примере:

```
$ pwd
/home/scott/libby
$ ls ~/pix
arrowrock.jpg   by_pool_03.jpg   on_floor_03.jpg
by_pool_01.jpg  on_floor_01.jpg  on_floor_04.jpg
by_pool_02.jpg  on_floor_02.jpg
$ ls
arrowrock.jpg   libby.jpg   libby_on_couch.jpg
$ cp ~/pix/on_floor_0[1-3].jpg
$ ls
arrowrock.jpg  libby_on_couch.jpg  on_floor_02.jpg
libby.jpg      on_floor_01.jpg     on_floor_03.jpg
```

Вывод подробной информации о копировании файлов

```
cp -v
```

Опция -v (или --verbose) предоставит вам подробную информацию о действиях, выполняемых командой cp.

```
$ pwd
/home/scott/libby
$ ls ~/pix
arrowrock.jpg   by_pool_03.jpg   on_floor_03.jpg
by_pool_01.jpg  on_floor_01.jpg  on_floor_04.jpg
by_pool_02.jpg  on_floor_02.jpg
$ ls
arrowrock.jpg   libby.jpg   libby_on_couch.jpg
$ cp -v ~/pix/on_floor_0[1-3].jpg .
'/home/scott/pix/on_floor_01.jpg' ->
  ./on_floor_01.jpg'
```

```
'/home/scott/pix/on_floor_02.jpg' -> '
  ./on_floor_02.jpg'
'/home/scott/pix/on_floor_03.jpg' -> '
  ./on_floor_03.jpg'
$ ls
arrowrock.jpg  libby_on_couch.jpg  on_floor_02.jpg
libby.jpg      on_floor_01.jpg     on_floor_03.jpg
```

Благодаря опции `-v` вы можете не задавать последнюю команду `ls`, так как в ваше распоряжение предоставляется информация, которая позволяет убедиться в том, что файлы скопированы.

Как предотвратить копирование поверх важных файлов

```
cp -i
```

Предыдущий пример продемонстрировал важную особенность команды `cp`, которую необходимо учитывать в процессе работы. В одном из более ранних примеров мы скопировали изображения `libby_on_floor` в каталог `libby`, в последнем примере мы снова выполнили копирование трех изображений `libby_on_floor` в тот же каталог `libby`. Несмотря на то что файлы с такими именами уже существовали в каталоге, мы не получили предупреждающего сообщения. Это общий принцип работы Linux: система предполагает, что пользователь знает, что он делает. При перезаписи существующих файлов система не предупредит вас ...если вы специально не попросите ее об этом. Если вы хотите, чтобы команда `cp` оповещала вас, перед тем, как записать один файл поверх другого, используйте опцию `-i` (или `--interactive`). Если вы снова попытаетесь скопировать те же файлы, но укажете опцию `-i`, результаты будут не такими, как раньше.

```
$ pwd
/home/scott/libby
$ ls ~/pix
arrowrock.jpg    by_pool_03.jpg    on_floor_03.jpg
by_pool_01.jpg   on_floor_01.jpg   on_floor_04.jpg
by_pool_02.jpg   on_floor_02.jpg
$ ls
arrowrock.jpg  libby_on_couch.jpg  on_floor_02.jpg
libby.jpg      on_floor_01.jpg     on_floor_03.jpg
$ cp -i ~/pix/on_floor_0[1-3].jpg .
cp: overwrite './on_floor_01.jpg'?
```

Как видите, команда `cp` приостановила свое выполнение, чтобы спросить вас, действительно ли вы хотите записать файл `libby_on_floor_01.jpg` поверх другого с таким же именем. Если вы хотите, чтобы файл был скопирован, введите `y`, в противном случае введите `n`. Если вы введете `n`, команда `cp` не прекратит свое выполнение: вам будет предложено подтвердить копирование очередного фала. Есть способ отменить копирование всех последующих файлов. Для этого надо нажать комбинацию клавиш <Ctrl+C>, завершив таким образом выполнение команды. Способа ответить `y` одновременно на все последующие вопросы нет. Таким образом, если вы захотите записать 1000 файлов поверх 1000 существующих, имеющих те же имена, и укажете опцию `-i`, вам придется запастись временем и терпением, так как система ровно 1000 раз спросит у вас, действительно ли вы хотите заменить существующий файл новым.

Внимание!

У рядовых пользователей редко возникает необходимость в опции `-i`. Для пользователя `root` она очень важна, так как его работа постоянно сопряжена с опасностью случайно заменить важный системный файл. По этой причине пользователю `root` желательно создать

в файле .bashrc псевдоним, наличие которого приведет к вызову cp -i вместо обычной команды cp.

```
alias cp='cp -i'
```

Копирование каталогов

```
cp -R
```

До сих пор мы рассматривали копирование файлов, но нередко приходится копировать целые каталоги. При этом команда cp *исходный_каталог целевой_каталог* работать не будет.

```
$ pwd
/home/scott
$ cp libby libby_bak
cp: omitting directory 'libby'
```

Для копирования каталогов надо использовать опцию -R (или --recursive), уже знакомую вам по команде ls. Опция -R, указанная при вызове команды cp, приводит к копированию каталога и его содержимого.

```
$ pwd
/home/scott
$ ls -l
drwxr-xr-x   328  2005-10-17  14:42  documents
drwxr-xr-x   240  2005-10-20  17:16  libby
$ cp -R libby libby_bak
$ ls -l
drwxr-xr-x   328  2005-10-17  14:42  documents
drwxr-xr-x   240  2005-10-20  17:16  libby
drwxr-xr-x   240  2005-10-20  17:17  libby_bak
```

Использование команды cp для создания резервных копий

```
cp -a
```

Когда вы изучите команду cp, вам, возможно, покажется, что она хорошо подходит для создания резервных копий. Действительно, с ее помощью можно решать эту задачу, однако существуют инструменты, гораздо более пригодные для этого (о них вы узнаете в главе 15). С помощью нескольких строк сценария оболочки bash можно обеспечить копирование различных файлов и каталогов. Чаще всего для этой цели используется опция -a (или --archive), которая эквивалента сочетанию трех опций -dpR (или --no-dereference --preserve --recursive). Действие опции -a можно описать так: cp не обрабатывает символьные ссылки (в противном случае объем копируемой информации мог бы неконтролируемо увеличиться), сохраняет основные атрибуты файла, например сведения о владельце, дату и время, и рекурсивно обрабатывает подкаталоги.

```
$ pwd
/home/scott
$ ls -l
drwxr-xr-x  216  2005-10-21  11:31  libby
drwxr-xr-x  216  2005-09-29  23:17  music
$ ls -lR libby
libby:
total 312
-rw-r--r--  73786  2005-10-20 12:12 arrowrock.jpg
-rw-r--r--  18034  2005-04-19 00:57 libby.jpg
-rw-r--r-- 198557  2005-04-19 00:57 on_couch.jpg
drwxr-xr-x    168  2005-10-21 11:31 on_floor
```

```
libby/on_floor:
total 764
-rw-r--r-- 218849 2005-10-20 16:11 on_floor_01.jpg
-rw-r--r-- 200024 2005-10-20 16:11 on_floor_02.jpg
-rw-r--r-- 358986 2005-10-20 16:11 on_floor_03.jpg
$ cp -a libby libby_bak
$ ls -l
drwxr-xr-x    216 2005-10-21 11:31 libby
drwxr-xr-x    216 2005-10-21 11:31 libby_bak/
drwxr-xr-x    216 2005-09-29 23:17 music
$ ls -lR libby_bak
libby:
total 312
-rw-r--r--  73786 2005-10-20 12:12 arrowrock.jpg
-rw-r--r--  18034 2005-04-19 00:57 libby.jpg
-rw-r--r-- 198557 2005-04-19 00:57 on_couch.jpg
drwxr-xr-x    168 2005-10-21 11:31 on_floor

libby/on_floor:
total 764
-rw-r--r-- 218849 2005-10-20 16:11 on_floor_01.jpg
-rw-r--r-- 200024 2005-10-20 16:11 on_floor_02.jpg
-rw-r--r-- 358986 2005-10-20 16:11 on_floor_03.jpg
```

На заметку

Если вы еще не поняли, почему я использую именно такие имена файлов, поясню: Libby — это моя собака. Во время написания книги она все время крутилась где-то рядом.

Как и при использовании команды cp, можно задавать текущий каталог с помощью точки.

```
$ pwd
/home/scott/libby
$ ls
arrowrock.jpg  libby.jpg  on_couch.jpg  on_floor
$ ls ~/pictures/dogs
on_floor_01.jpg  on_floor_03.jpg
on_floor_02.jpg  on_floor_04.jpg
$ mv ~/pictures/dogs/on_floor_04.jpg .
$ ls
arrowrock.jpg  on_couch.jpg  on_floor_04.jpg
libby.jpg      on_floor
$ ls ~/pictures/dogs
on_floor_01.jpg  on_floor_02.jpg  on_floor_03.jpg
```

Если вы перемещаете файл в другой каталог и хотите, чтобы он сохранил свое имя, вам надо задать только имя каталога. При этом имя файла останется прежним.

```
$ pwd
/home/scott/libby
$ ls
arrowrock.jpg  on_couch.jpg  on_floor_04.jpg
libby.jpg      on_floor
$ ls on_floor
on_floor_01.jpg  on_floor_02.jpg  on_floor_03.jpg
$ mv on_floor_04.jpg on_floor
$ ls
arrowrock.jpg  on_couch.jpg  on_floor_04.jpg
libby.jpg      on_floor
$ ls on_floor
on_floor_01.jpg  on_floor_03.jpg
on_floor_02.jpg  on_floor_04.jpg
```

дело обстоит по-другому. Как можно заметить из предыдущего примера, она, будучи вызвана без дополнительных опций, успешно перемещает или переименовывает каталоги.

Внимание!

Команда `mv` имеет одну важную особенность, которую начинающие пользователи часто упускают из виду. Если вы перемещаете ссылку, указывающую на каталог, вам надо внимательно следить за вводимыми данными. Предположим, что в вашем рабочем каталоге есть ссылка `dogs`, которая указывает на каталог `/home/scott/pictures/dogs`, и вы хотите переместить эту ссылку в каталог `/home/scott/libby`. Приведенная ниже команда перемещает лишь саму ссылку.

```
$ mv dogs ~/libby
```

Следующая же команда перемещает каталог, на который эта ссылка указывает:

```
$ mv dogs/ ~/libby
```

Различие между ними лишь в косой черте, которая вводится после ссылки. Если символ / отсутствует, перемещается лишь сама ссылка. Включив этот символ, вы переместите каталог, а не ссылку. Будьте внимательны!

Удаление файлов

`rm`

Команда `rm` (в ней нашли место только две буквы из слова "remove") безвозвратно удаляет файлы. В системе Linux нет "мусорной корзины". Одно неосторожное движение — и вернуть данные уже нельзя.

Может быть, не все так мрачно? Действительно, при работе с командной строкой "мусорная корзина" не используется, но кое-что все же сделать можно. Если вы остановите машину в тот момент, когда обнаружите ошибку, и если

Для того чтобы гарантировать, что on_floor — это каталог, желательно указать после него косую черту. Тогда команда примет следующий вид: mv libby_on_floor_04.jpg on_floor/. Если окажется, что on_floor — не каталог, команда не переместит файл. Так можно предотвратить запись одного файла поверх другого.

На заметку

Многие опции команд cp и mv совпадают и производят одинаковые действия. Например, опция -i запрашивает подтверждение на выполнение операции, а если указана опция -v, то при копировании и перемещении выводятся подробные сведения о выполненных действиях.

Переименование файлов и каталогов

mv

Как вы вскоре увидите, команда mv может делать нечто большее, чем обычное перемещение файлов. Она позволяет также переименовывать файлы. При перемещении файла указывается целевое имя, которое не обязательно совпадает с именем исходного файла. Именно это свойство пользователи, работающие с оболочкой, издавна используют для переименования файлов и каталогов.

```
$ pwd
/home/scott/libby/by_pool
$ ls -F
libby_by_pool_02.jpg  liebermans/
$ mv liebermans/ lieberman_pool/
$ ls -F
libby_by_pool_02.jpg  lieberman_pool/
```

Копируя каталог с помощью команды cp, необходимо указывать опцию -R (или --recursive). С командой mv

Перемещение и переименование файлов

mv

Как вам уже известно, команда cp копирует файлы. А можно ли не копировать, а лишь переместить файл? Для этой цели используется команда mv. Подобно команде cp, ее имя получается путем удаления гласных из слова "move".

Прочитав описание mv, нетрудно заметить, что для нее допустимы многие из опций, известных вам по команде cp. Это неудивительно, ведь mv делает то же, что и cp -a, а после этого, если копирование было произведено успешно, удаляет исходный файл.

Самое простое описание mv звучит так: эта команда перемещает файл из одной позиции файловой системы в другую.

```
$ pwd
/home/scott/libby
$ ls
libby_arrowrock.jpg  libby_bak.jpg  libby.jpg
libby_on_couch.jpg   on_floor
$ ls ~/pictures/dogs
libby_on_floor_01.jpg  libby_on_floor_03.jpg
libby_on_floor_02.jpg  libby_on_floor_04.jpg
$ mv ~/pictures/dogs/libby_on_floor_04.jpg
  libby_on_floor_04.jpg
$ ls
libby_arrowrock.jpg    libby.jpg
libby_on_floor_04.jpg  libby_bak.jpg
libby_on_couch.jpg     on_floor
$ ls ~/pictures/dogs
libby_on_floor_01.jpg   libby_on_floor_02.jpg
libby_on_floor_03.jpg
```

операционная система еще не записала данные в те сектора, в которых содержался удаленный файл, и если вы сможете умело воспользоваться сложными инструментами восстановления информации, есть шанс вернуть файл. Но в любом случае это займет много времени. Поэтому будьте внимательны.

Совет

Попытки обезопасить команду rm предпринимаются многими. Некоторые заменяют ее командой перезаписи во временное хранилище (http://www.comwt.com/open_source/projects/trashcan/), другие реализуют в оболочке поддержку "мусорной корзины" (http://pages.stern.nyu.edu/~marriaga/software/libtrash/). Известно даже о замене rm новой командой trash (http://www.ms.unimelb.edu.au/~jrlooker/shell.html#trash).

С другой стороны, иногда хочется быть уверенным, что никто, даже специалист высочайшей квалификации, не сможет восстановить удаленные файлы. В этом случае вместо rm желательно использовать команду shred. Эта команда 25 раз перезаписывает область на диске, где хранился файл, так, что прочитать данные оказывается невозможно.

Команда rm проста в использовании, можно даже сказать, слишком проста.

```
$ pwd
/home/scott/libby/by_pool/lieberman_pool
$ ls
pool_01.jpg         pool_03.jpg
pool_01.jpg_bak     pool_03.jpg_bak
$ rm pool_01.jpg_bak
$ ls
pool_01.jpg    pool_03.jpg    pool_03.jpg_bak
```

Удаление нескольких файлов с помощью символов групповых операций

```
rm *
```

Символы групповых операций, например *, позволяют одним нажатием клавиши удалить несколько файлов.

```
$ pwd
/home/scott/libby/by_pool/lieberman_pool
$ ls
pool_01.jpg       pool_03.jpg
pool_01.jpg_bak   pool_03.jpg_bak
$ rm *_bak
$ ls
pool_01.jpg  pool_03.jpg
```

Внимание!

Будьте предельно внимательны, удаляя файлы с указанием символов групповых операций, потому что вы можете удалить гораздо больше, чем собирались. Классический пример — ввод вместо `rm *txt` команды `rm * txt`. Вместо текстовых файлов будут удалены все файлы, а затем команда предпримет попытку удалить файл `txt`.

Вывод подробной информации при удалении файлов

```
rm -v
```

Если вы хотите получать сведения о всех действиях, выполняемых командой `rm`, используйте опцию `-v` (или `--verbose`).

```
$ pwd
/home/scott/libby/by_pool/lieberman_pool
$ ls
pool_01.jpg       pool_03.jpg
pool_01.jpg_bak   pool_03.jpg_bak
$ rm -v *_bak
removed 'pool_01.jpg_bak'
removed 'pool_03.jpg_bak'
$ ls
pool_01.jpg  pool_03.jpg
```

Как предотвратить удаление важных файлов

```
rm -i
```

Опция `-i` (или `--interactive`) делает команду `rm` более безопасной. В этом случае команда запрашивает у пользователя подтверждение на удаление каждого файла. Эта опция незаменима при работе с правами root!

```
$ pwd
/home/scott/libby/by_pool/lieberman_pool
$ ls
pool_01.jpg       pool_03.jpg
pool_01.jpg_bak   pool_03.jpg_bak
$ rm -i *_bak
rm: remove regular file 'pool_01.jpg_bak'? y
rm: remove regular file 'pool_03.jpg_bak'? y
$ ls
pool_01.jpg  pool_03.jpg
```

Как и в командах, рассмотренных ранее, ответ `y` на запрос команды означает согласие на удаление файла, а ответ `n` — приказ сохранить файл. Получив ответ `n`, команда не прекращает работу, а переходит к обработке следующего файла.

Удаление пустого каталога

```
rmdir
```

Удалить файл несложно, но что делать с каталогами?

```
$ pwd
/home/scott/libby/by_pool
$ ls
pool_02.jpg  lieberman_pool  lieberman_pool_bak
$ ls lieberman_pool_bak
pool_01.jpg       pool_03.jpg
pool_01.jpg_bak   pool_03.jpg_bak
$ rm lieberman_pool_bak
rm: cannot remove 'lieberman_pool_bak/':
  Is a directory
```

Действительно, попытка не удалась. Однако существует команда `rmdir`, специально предназначенная для удаления каталогов. Попробуем использовать ее.

```
$ rmdir lieberman_pool_bak
rmdir: 'lieberman_pool_bak/': Directory not empty
```

И опять неудача. Команда `rmdir` может удалить только пустой каталог. В нашем случае в каталоге `lieberman_pool_bak` содержатся только четыре файла, так что нетрудно удалить их, а затем использовать `rmdir`. Но что делать, если надо удалить каталог, содержащий 10 подкаталогов. Более того, в каждом из них находится 10 подкаталогов, каждый из которых содержит 25 файлов? Работа по удалению каталогов затянется надолго. Однако есть более простой путь.

Удаление файлов и каталогов, содержащих данные

```
rm -Rf
```

Для решения этой задачи надо использовать сочетание опций `-R` (или `--recursive`) и `-f` (или `--force`). Опция `-R` указывает команде `rm` на то, что надо перейти в каждый подкаталог и удалить его содержимое, а опция `-f` говорит, что пользователя не следует беспокоить напоминанием о том, что очередной каталог не пустой.

```
$ pwd
/home/scott/libby/by_pool
$ ls
pool_02.jpg  lieberman_pool  lieberman_pool_bak
$ ls lieberman_pool_bak
pool_01.jpg       pool_03.jpg
pool_01.jpg_bak   pool_03.jpg_bak
$ rm -Rf lieberman_pool_bak
$ ls
pool_02.jpg  lieberman_pool
```

Это удобный способ избавиться от каталога и всех его подкаталогов.

Внимание!

Команда `rm -Rf` может удалить важные файлы и разрушить саму операционную систему.

Классическое предупреждение пользователям Linux: не используйте команду `rm -Rf /*`, если вы работаете как пользователь `root`. Так можно стереть всю систему. Пользователь, сделавший это, выглядит довольно глупо.

В любом случае, используя символы групповых операций в команде `rm -Rf`, надо соблюдать осторожность. Различие между командами `rm -Rf libby*` и `rm -Rf libby *` чрезвычайно велико. Первая из

них удаляет в рабочем каталоге все файлы, начинающиеся на libby; вторая команда сначала удаляет файл libby, а затем и все остальные файлы в каталоге и его подкаталогах.

Если вместо команды rm -Rf ~/libby/* вы зададите rm -Rf ~/ libby /*, у вас возникнут большие проблемы. Сначала исчезнет каталог ~/libby, а затем вся файловая система.

И еще одно предупреждение пользователям: никогда не задавайте команду rm -Rf .*/*, чтобы удалить файлы, начинающиеся с точки, поскольку указанному критерию удовлетворяет каталог . ., и вы удалите все данные, расположенные выше вашего рабочего каталога.

Еще раз запомните: используя rm -Rf, соблюдайте осторожность! Удвойте внимание, если вы работаете с полномочиями root!

Проблемы при удалении файлов

Прежде чем закончить разговор о команде rm, обсудим некоторые особенности, связанные с применением этой команды к вашей файловой системе. Во-первых, как бы вы ни старались, вы не сможете удалить каталог . или . ., так как он необходим для поддержки самой файловой системы. Во-вторых, зачем их удалять? Оставим их в покое!

Как удалить файл, в имени которого содержится пробел? Обычный способ — вызов команды rm и указание имени файла в качестве параметра — не подходит, так как команда интерпретирует заданное имя как два отдельных параметра. На самом деле решить эту задачу несложно. Достаточно поместить имя файла в кавычки.

```
$ ls
cousin harold.jpg -cousin_roy.jpg cousin_beth.jpg
$ rm cousin harold.jpg
rm: cannot remove 'cousin':
  No such file or directory
rm: cannot remove 'harold.jpg':
```

```
  No such file or directory
$ rm "cousin harold.jpg"
$ ls
-cousin_roy.jpg  cousin_beth.jpg
```

И еще одна проблема: как удалить файл, имя которого начинается с дефиса?

```
$ ls
-cousin_roy.jpg  cousin_beth.jpg
$ rm -cousin_roy.jpg
rm: invalid option -- c
Try 'rm -help' for more information.
```

Команда `rm` воспринимает символ - как признак опции, но в данном случае не распознает опцию, начинающуюся с буквы c, за которой следуют символы `ousin_roy.jpg`. В результате команда не знает, как поступить.

Существуют два решения этой проблемы. Во-первых, вы можете предварить имя файла двумя дефисами (`--`). Это означает, что следующие за ними данные должны восприниматься не как опция, как имя файла или каталога.

```
$ ls
-cousin_roy.jpg  cousin_beth.jpg
$ rm -- -cousin_roy.jpg
$ ls
cousin_beth.jpg
```

Во-вторых, вы можете использовать точку как часть пути к файлу и тем самым устранить тот самый пробел перед дефисом, который ввел в заблуждение команду `rm`.

```
$ ls
-cousin_roy.jpg  cousin_beth.jpg
$ rm ./-cousin_roy.jpg
$ ls
cousin_beth.jpg
```

Этот пример еще раз подтверждает: изобретательность пользователей Linux не знает границ. Однако лучше всего не включать дефис в начало файла!

Как превратиться в другого пользователя

```
su имя_пользователя
```

Команда su (сокращение от switch user — переключение пользователя, и, вопреки бытующему мнению, отнюдь не super user — суперпользователь) позволяет одному пользователю временно действовать от имени другого. Чаще всего она используется тогда, когда необходимо быстро получить полномочия root, выполнить одну-две команды, а затем продолжить работу от имени обычного пользователя.

Выполнить команду su несложно. Достаточно ввести ее имя, а затем указать пользователя, от имени которого вы собираетесь выступить.

```
$ ls
/home/scott/libby
$ whoami
scott
$ su gromit
Password:
$ whoami
gromit
$ ls
/home/scott/libby
```

В этом примере встретилась новая команда: whoami. Она используется нечасто и сообщает пользователю, кем он является с точки зрения оболочки. В данном случае мы применяем ее для того, чтобы убедиться, что команда su работает именно так, как мы того ожидаем.

Как превратиться в другого пользователя и использовать его переменные окружения

```
su -l
```

Команда su будет выполнена только в том случае, если вам известен пароль пользователя. Если вы не зададите пароль, превращение не состоится. При нормальном выполнении команды вы продолжите работу с той оболочкой, которая была задана для целевого пользователя в файле /etc/passwd, например sh, tcsh или bash. Большинство пользователей Linux предпочитают оболочку bash, поэтому, вероятнее всего, вы не почувствуете разницы. В предыдущем примере работа даже будет продолжена с тем же каталогом. Другими словами, вы станете пользователем gromit, но по-прежнему будете работать с переменными окружения, установленными для пользователя scott. Представьте себе, что вы одели одежду чемпиона мира и соответствующим образом загримировались. Вы будете выглядеть в точности как он, но не будете обладать его мастерством.

Справиться с данной проблемой поможет опция -l (или --login).

```
$ ls
/home/scott/libby
$ whoami
scott
$ su -l gromit
Password:
$ whoami
gromit
$ ls
/home/gromit
```

На первый взгляд, все происходит практически так же, как и в предыдущем примере, но на самом деле различия

огромны. Уже тот факт, что вы находитесь в рабочем каталоге gromit, говорит о том, что что-то изменилось. Опция -l сообщает su о необходимости использовать оболочку, предусмотренную в процедуре регистрации, т.е. поступать так, как будто gromit на самом деле прошел регистрацию в системе. Теперь вы не только носите имя gromit, вы используете его переменные окружения, находитесь в его рабочем каталоге, одним словом, чувствуете себя так же, как gromit после регистрации. Продолжая сравнение, приведенное выше, теперь вы не только выглядите как чемпион мира, но и способны устанавливать рекорды!

Как превратиться в пользователя root

```
su
```

Вы уже знаете, что команда su чаще всего используется для временного получения полномочий root. Для решения этой задачи вы можете задать команду su root или, еще лучше, su -l root, однако существует более быстрый способ.

```
$ whoami
scott
$ su
Password:
$ whoami
root
```

Как стать пользователем root и использовать его переменные окружения

```
su -
```

Команда su эквивалента команде su root. Выполнив ее, вы получаете полномочия root, но этим ваши возмож-

ности исчерпываются. Вы по-прежнему работаете с переменными окружения другого пользователя.

```
$ ls
/home/scott/libby
$ whoami
scott
$ su
Password:
$ whoami
root
$ ls
/home/scott/libby
```

Добавив дефис после команды `su`, вы получите тот же эффект, что и от выполнения команды `su -l root`.

```
$ ls
/home/scott/libby
$ whoami
scott
$ su -
Password:
$ whoami
root
$ ls
/root
```

Теперь вы носите имя `root` и работаете с его переменными окружения, короче говоря, становитесь полноценным суперпользователем. Все, что может сделать пользователь `root`, доступно также и вам. Можете наслаждаться неограниченными возможностями, но помните, чем кончается незначительная на первый взгляд ошибка.

Выводы

Если бы эта книга посвящалась не операционной системе, а аппаратному обеспечению, мы рассматривали бы в ней оперативную память, жесткие диски и материнские платы. Однако мы все же изучаем Linux, поэтому речь здесь идет о командах `ls`, `pwd`, `cd`, `touch`, `mkdir`, `cp`, `mv`, `rm` и `su`, которые должен знать любой пользователь, чтобы эффективно работать с командной строкой. Мы также вскользь коснулись команд `rmdir` и `whoami`. Теперь вы готовы к тому, чтобы ...продолжить изучение команд.

3

Получение информации о командах

В предыдущей главе мы начали изучать основные команды системы Linux. Мы рассмотрели несколько из них, но осталось гораздо больше. Команда `ls` — мощный инструмент, поддерживающий огромное количество опций, гораздо больше, чем было рассмотрено в главе 2. Как же получить дополнительную информацию об этой команде и других, интересующих вас? И как разобраться в командах, если неизвестны даже их имена? В этом вам поможет материал данной главы. Вы узнаете, как глубже изучить те команды, о которых вы успели составить представление, те, о которых кроме имени вы не знаете ничего, и даже полностью неизвестные команды.

Начнем с "гигантов" — команд `man` и `info`, а затем перейдем к менее глобальным командам, которые тем не менее используют ту же информацию, что и команда `man`. Дойдя до конца этой главы, вы будете готовы продолжить изучение самых разнообразных средств, доступных посредством оболочки.

Получение информации о командах с помощью команды man

```
man ls
```

Хотите получше изучить какую-либо из команд Linux? Нет ничего проще. Предположим, вам понадобилась информация о команде `ls`. Введите в командной строке `man ls`, и на экране появится страница справочного руководства (сама команда `man` сокращенно означает manual, т.е. руководство), на которой подробно описываются все особенности команды `ls`. В качестве параметра `man` можно указать имя любой команды. Все они (или большинство из них) отражены в справочной системе.

Несмотря на то что справочное руководство чрезвычайно полезно, при работе с командой `man` подчас возникают проблемы. Необходимо знать имя команды (хотя эту трудность можно обойти), кроме того, нередко информация оказывается устаревшей и не отражает средства, появившиеся относительно недавно. О некоторых командах сведения вовсе отсутствуют. Однако хуже всего, когда описание интересующей вас команды имеется в наличии и было создано недавно, но, ознакомившись с ним, вы обнаруживаете, что оно практически бесполезно.

Обычно страницы справочного руководства разрабатывают те же специалисты, которые пишут программы (бывают и исключения, но, как правило, это так). Большинство разработчиков, приложения которых были включены в дистрибутив Linux, являются первоклассными программистами, но они не всегда могут доходчиво описать работу программы. Они прекрасно разбираются в предмете, но слишком часто забывают о том, что то, что очевидно для разработчиков, зачастую неизвестно пользователям.

Несмотря на перечисленные выше проблемы, справочное руководство — незаменимый ресурс для пользователей Linux всех уровней квалификации. Если вы хотите работать с командной строкой, вам надо научиться пользоваться руководством. Как было сказано ранее, использовать команду man несложно. Надо ввести man, а затем указать имя команды, о которой вы хотите получить дополнительные сведения.

```
$ man ls
LS(1)            User Commands           LS(1)
NAME
  ls - list directory contents
SYNOPSIS
  ls [OPTION]... [FILE]...
DESCRIPTION
  List information about the FILEs (the current
  directory by default).
  Sort entries alphabetically if none of -cftuSUX
  nor --sort.
  Mandatory arguments to long options are
  mandatory for short options too.
  -a, --all
    do not hide entries starting with .
  -A, --almost-all
    do not list implied . and ..
[Информация сокращена для экономии места]
```

Объем данных, предоставляемых man, часто бывает очень велик. Для данного примера он составляет 218 строк. Описание некоторых команд очень короткое, однако есть и такие команды, описание которых гораздо длиннее, чем приведенное в данном примере. Ваша задача — ознакомиться с описанием, которое в большинстве случаев включает в себя следующие разделы.

- NAME. Имя команды и краткое описание.
- SYNOPSIS. Формат вызова команды.
- DESCRIPTION. Подробное описание возможностей, предоставляемых командой.
- OPTIONS. Чаще всего именно этот раздел оказывается самым полезным для пользователей. В нем приводятся опции, предусмотренные для команды, и их краткое описание.
- FILES. Файлы, используемые командой.
- AUTHOR. Автор, который разработал программу, реализующую эту команду; контактная информация.
- BUGS. Замеченные недостатки и сведения о том, куда сообщить, если вы обнаружите новую ошибку.
- COPYRIGHT. Информация об авторских правах.
- SEE ALSO. Другие команды, имеющие отношение к рассматриваемой.

Просмотр страницы справочной системы происходит следующим образом. Для того чтобы перейти на строку вниз, следует нажать клавишу со стрелкой вниз, чтобы переместиться на одну строку вверх, надо нажать клавишу со стрелкой вверх. Перейти вперед на одну страницу позволяет клавиша <F> (первая буква в слове forward), а на страницу назад — клавиша <B> (первая буква в слове back). Когда вы достигнете конца страницы, выполнение команды man автоматически завершится и вы вернетесь в среду оболочки. В некоторых случаях автоматическое завершение не происходит, тогда надо нажать клавишу <Q>, чтобы выйти из программы. Завершить команду можно в любой момент с помощью клавиши <Q>; при этом не обязательно находиться в конце страницы.

Нередко бывает нелегко найти требуемые сведения на странице, поэтому пользователю предоставляются мини-

мальные средства поиска. Для того чтобы отыскать данные на открытой странице справочного руководства, надо ввести косую черту, затем термин, который надо найти, и нажать клавишу <Enter>. Если этот термин существует, содержимое страницы сдвинется так, чтобы он отображался на экране. Чтобы продолжить поиск, надо повторно нажимать клавишу <Enter> (или <N>). Чтобы вернуться к предыдущему вхождению термина, надо нажать комбинацию клавиш <Shift+N>.

Поиск команды по выполняемым ею действиям

```
man -k
```

Немного поработав с командой man, вы обнаружите, что она предоставит нужную вам информацию ... при условии, что вы знаете, на какой странице эта информация находится. Но что делать, если вам известно, какие действия должна выполнять команда, но вы не знаете ее реального имени? В этой ситуации вам поможет опция -k (или --apropos). Она позволяет организовать поиск по слову или фразе, описывающей команду. На экран выводится список команд, для которых либо имя совпадает с ключевым словом, либо это слово содержится в разделе SYNOPSIS.

```
$ man list
No manual entry for list
$ man -k list
last (1) - show listing of last logged in users
ls (1) - list directory contents
lshal (1) - List devices and their properties
lshw (1) - list hardware
lsof (8) - list open files
[Информация сокращена для экономии места]
```

Опцию -k надо применять осторожно, так как объем выходных данных может быть очень большим и вы не сможете найти в них те сведения, которые вам нужны. Если поиск закончился безрезультатно, попробуйте повторить его, задав другое ключевое слово.

Совет

Опция -k или ее полный вариант, --apropos, делает то же, что и команда apropos, которая будет рассмотрена далее в этой главе.

Получение кратких сведений о команде

```
man -f
```

Если вам известно имя команды, но вы не знаете, какие действия она выполняет, с ее назначением можно быстро ознакомиться, не просматривая всю страницу справочного руководства. Если вы зададите опцию -f (или --whatis), на экране отобразится раздел SYNOPSIS.

```
$ man -f ls
ls (1)               - list directory contents
```

Совет

Опция -f, или -whatis, — это аналог команды whatis, которая будет рассмотрена несколько позже.

Формирование базы данных команд

```
man -u
```

Иногда бывает, что при попытке получить информацию о команде посредством man вы получаете сообщение, что страница для этой команды отсутствует. Не спешите сми-

риться с подобным положением вещей; попытайтесь выполнить команду man с опцией -u (или --update). Эта опция вызывает принудительное формирование базы данных команд, которую использует команда man.

```
$ man ls
No manual entry for ls
$ man -u ls
LS(1)          User Commands          LS(1)
NAME
  ls - list directory contents
SYNOPSIS
  ls [OPTION]... [FILE]...
[Информация сокращена для экономии места]
```

Просмотр страницы справочной системы, посвященной конкретной команде

```
man [1-8]
```

В предыдущем листинге вы, вероятно, заметили, что на странице для команды ls отображается последовательность символов LS(1). Ранее, когда мы рассматривали опцию -k, имя каждой команды сопровождалось числом в скобках. В большинстве случаев после имени отображалось число 1, но после команды lsof было выведено число 8. Что же означают эти цифры?

Страницы справочного руководства распределены по разделам с номерами от 1 до 8. Назначение разделов описано ниже (если какой-то из примеров будет непонятен вам, не беспокойтесь; большинство команд узкоспециализированы).

1. Команды общего назначения, например cd, chmod, lp, mkdir, passwd.

2. Низкоуровневые системные вызовы, поддерживаемые ядром, например intro, chmod.
3. Функции библиотеки C, например beep, HTML::Parser, Mail::Internet.
4. Специальные файлы, к ним, в частности, относятся устройства в каталоге /dev, например console, lp, mouse.
5. Информация о форматах файлов и соглашениях, например apt.conf, dpkg.cfg, hosts, passwd.
6. Игры, например atlantik, bouncingcow, kmahjongg, rubik.
7. Различная информация, в том числе сведения о макропакетах, например ascii, samba, utf-8.
8. Команды системного администрирования, вызываемые пользователем root, например mount, shutdown.

До сих пор все интересовавшие нас команды располагались в разделе 1. Это неудивительно, так как мы изучаем общие вопросы работы с системой Linux. Заметьте, что некоторые ключевые слова соответствуют нескольким разделам, например, информация о chmod находится в разделах 1 и 2, а сведения о passwd можно получить, обратившись к разделу 1 или 5. По умолчанию принимается наименьший из возможных номеров. Так, если вы введете в командной строке man passwd, то получите информацию о команде с таким именем из раздела 1. Если вас интересует файл passwd, полученная информация вряд ли окажется полезной для вас. Для того чтобы получить страницу для файла passwd, укажите после man требуемый номер раздела.

```
$ man passwd
PASSWD(1)                                    PASSWD(1)
NAME
  passwd - change user password
SYNOPSIS
  passwd [-f|-s] [name]
  passwd [-g] [-r|-R] group
  passwd [-x max] [-n min] [-w warn] [-i inact]
login
  passwd {-l|-u|-d|-S|-e} login
DESCRIPTION
 passwd changes passwords for user and group
 accounts. A normal user...
[Информация сокращена для экономии места]
$ man 5 passwd
PASSWD(5)                                    PASSWD(5)
NAME
  passwd - The password file
DESCRIPTION
  passwd contains various pieces of information
 for each user account.
[Информация сокращена для экономии места]
```

Вывод справочной информации на печать

```
man -t
```

Просматривать справочную информацию на экране терминала несложно, однако в некоторых случаях необходимо выводить ее на печать. Процесс вывода на печать включает несколько этапов; команды, встречающиеся в этом разделе, используют принципы, которые мы подробнее рассмотрим несколько позже. Если вы хотите вывести страницу справочной информации на принтер, достаточно лишь

воспользоваться предлагаемыми здесь командами. Когда вы прочитаете последующие главы, эти команды станут для вас более понятными.

Предположим, что у вас есть принтер, идентифицируемый именем `hp_laserjet`, который присоединен к вашей системе. Если вам надо напечатать страницу справочного руководства, посвященную команде `ls`, используйте опцию `-t` (или `--troff`), а затем средствами конвейерной обработки передайте выходные данные команде `lpr`. В данном случае принтер определяется с помощью опции `-P`.

```
$ man -t ls | lpr -P hp_laserjet
```

На заметку

О символе `|`, посредством которого задается конвейерная обработка, вы узнаете из главы 4, а команда `lpr` будет рассмотрена в главе 6.

Практически сразу же или с небольшой задержкой, в зависимости от скорости вашего компьютера и принтера, `hp_laserjet` начнет выдавать бумажные листы с информацией о команде `ls`. Теперь представьте себе, что вам не надо печатать страницу справочного руководства, а достаточно создать документ в формате PDF. Команды, решающие эту задачу, могут показаться непонятными, но по мере чтения материала данной книги все прояснится.

В данном случае также используется опция `-t`, однако на этот раз данные передаются в файл в формате PostScript. Если данный процесс завершится успешно, PostScript-файл будет преобразован в формат PDF с помощью команды `ps2pdf`, и если это преобразование также будет успешным, исходный PostScript-файл будет удален, так как он больше не нужен.

```
$ man -t ls > ls.ps && ps2pdf ls.ps && rm ls.ps
```

На заметку

О назначении символов > и && вы узнаете из главы 4, а команда ps2pdf будет рассмотрена в главе 6.

Теперь вы знаете, как создавать бумажные копии страниц справочного руководства, посвященных самым нужным для вас командам. Вы сможете также создать набор страниц в формате PDF, которые при необходимости можно легко вывести на принтер. Как видите, команда man очень мощная и гибкая и предоставляет такие возможности, о которых многие пользователи даже не догадываются.

Получение информации о командах с помощью info

```
info
```

Команда man проста в использовании, а предоставляемые ею страницы справочного руководства удобны, однако их нельзя назвать дружественными по отношению к пользователю. В рамках проекта GNU был разработан альтернативный формат — info-страницы, для доступа к которым используется команда info.

Многие пользователи считают, что info-страницы лучше написаны, чем страницы man, на них более полно представлена информация, однако страницы man проще в работе. Для конкретной команды страница man только одна, а содержимое страниц info организовано в виде разделов, называемых узлами, которые могут включать подразделы, называемые подузлами. Для того чтобы эффективно работать с системой info, надо освоить навигацию не только в пределах одной страницы, но также между узлами и подузлами. Поначалу бывает довольно трудно найти информацию на страницах info. Как ни странно, именно система,

разработанная специально для начинающих, оказывается более сложной в изучении.

Обращаться к системе info удобнее всего с помощью команды `info`. Таким образом, чтобы получить сведения о системе info, надо использовать следующую команду:

```
$ info info
```

В результате ее выполнения будет открыта info-страница для команды `info`. Попробуем научиться ориентироваться в пространстве info.

Навигация в системе info

В пределах одного раздела действуют следующие правила. Чтобы перейти вниз или вперед на одну строку, надо нажать клавишу со стрелкой вниз. Аналогично клавиша со стрелкой вверх позволяет перейти на одну строку вверх или назад. Когда вы достигнете конца раздела, курсор не будет реагировать на попытки сдвинуться вниз.

Если вы хотите переместиться вниз на один экран, надо использовать клавишу <PageDown>, а если на один экран вверх — клавишу <PageUp>. Описанные здесь клавиши не позволят покинуть текущий раздел.

Если вы достигли конца раздела и хотите перейти в его начало, нажмите клавишу <B> (первая буква в слове beginning).

Если, перемещаясь в пределах раздела, вы заметили, что информация выглядит странно, например, символы или слова искажены, вам надо нажать комбинацию клавиш <Ctrl+L>, чтобы обновить содержимое экрана.

Теперь, когда вы знаете, как перемещаться в пределах раздела или узла, рассмотрим навигацию между узлами. Если вы не хотите перемещаться вперед и назад с помощью клавиш <PageDown> и <PageUp>, можете использовать

для перемещения вниз клавишу пробела, а для перемещения вверх — клавишу <Backspace> или <Delete>. Действие этих клавиш несколько отличается от действия клавиш <PageDown> и <PageUp>: если вы достигнете конца раздела, то перейдете к следующему разделу (или подразделу, если таковой существует). Перемещаясь назад и достигнув начала раздела, вы перейдете к предыдущему разделу или подразделу. Используя пробел или клавиши <Backspace> и <Delete>, вы можете просмотреть весь набор info-страниц для конкретной команды.

Чтобы перейти к следующему разделу несколько быстрее и избежать многократных нажатий клавиш, вы можете воспользоваться клавишей <N> (next). Если раздел, который вы просматриваете, содержит подразделы, то, нажав <N>, вы пропустите подразделы и сразу перейдете к следующему разделу того же уровня, что и текущий. Если вы просматриваете подраздел и нажмете клавишу <N>, то перейдете к следующему подразделу. Подобно клавише <N>, вызывающей перемещение к следующему разделу того же уровня, клавиша <P> (previous) переместит вас к предыдущему разделу.

Если вам надо переместиться вперед к следующему элементу, независимо от того, является ли он узлом или подузлом, надо использовать клавишу <]>. Если вы просматриваете узел, в котором есть подузлы, то после нажатия клавиши <]> перейдете к первому из них, если же подузлов нет, то эта клавиша переместит вас к следующему узлу того же уровня. Чтобы переместиться назад подобным образом, надо использовать клавишу <[>.

Переместиться к вышестоящему или родительскому узлу позволяет клавиша <U> (up). При этом будьте осторожны: нажав данную клавишу лишний раз, вы подниметесь выше "корневого узла" для текущей команды и очутитесь в узле `Directory` — настоящем корневом узле, из которого

можно перейти к любому другому узлу info. (Достичь узла Directory можно также, нажав клавишу <D> (directory), причем сделать это можно в любой момент.)

Узел Directory представляет один из типов страниц, доступных посредством системы info, и именно, страницу Menu, в которой перечисляются подтемы текущей темы. Если вы окажетесь на странице Menu, то сможете переместиться к одному из указанных в нем подузлов одним из двух способов. Введите m (menu), а затем имя подузла, в который вам необходимо перейти. Например, ниже показана первая страница, которую вы увидите, выполнив команду info info.

```
File: info.info, Node: Top, Next: Getting
  Started, Up: (dir)

Info: An Introduction
*********************
The GNU Project distributes most of its on-line
manuals in the "Info format", which you read
using an "Info reader". You are probably using an
Info reader to read this now.
[Информация сокращена для экономии места]

* Menu:
* Getting Started:: Getting started using an
    Info reader.
* Expert Info:: Info commands for experts.
* Creating an Info File:: How to make your own
    Info file.
* Index:: An index of topics, commands, and
    variables.
```

Для того чтобы перейти к Expert Info, надо ввести букву m и за ней последовательность Exp. После этого можно

либо заканчивать ввод, т.е. задавать символы `ert Info`, либо нажать клавишу <Tab>, и система info автоматически сформирует имя пункта, соответствующего уже введенным символам. Если система info отказывается формировать имя пункта, это значит, что вы допустили ошибку либо введенным вам символам соответствует несколько пунктов меню. Исправьте ошибку или продолжайте ввод до тех пор, пока info сможет однозначно идентифицировать интересующий вас пункт. Если в этот момент окажется, что вам не надо покидать данную страницу, нажмите комбинацию клавиш <Ctrl+G>, чтобы отменить команду, и продолжайте просмотр текущего узла.

Существует и альтернативный способ. Вы можете, используя клавиши со стрелкой вверх или вниз, разместить курсор на требуемом пункте меню и нажать клавишу <Enter>.

Если вам надо осуществить не навигацию, а поиск, у вас также есть два способа сделать это. Вы можете искать по заголовкам узлов либо по всему тексту. Для того чтобы искать в заголовках, введите `i` (index), затем ключевое слово и нажмите клавишу <Enter>. Если слово присутствует в заголовке, вы можете перейти к соответствующему узлу. Можно также повторить поиск и перейти к следующему результату, для этого надо ввести запятую.

Если вы хотите выполнить поиск во всем тексте, введите `s` (search), затем ключевое слово или фразу и нажмите клавишу <Enter>. Для того чтобы повторить поиск, надо непосредственно после нажатия клавиши <Enter> нажать клавишу <S>. Сделать это сложнее, чем ввести запятую, но такой подход позволяет добиться нужных результатов.

Если вы "заблудились" в системе info и вам нужна помощь, нажмите клавишу <?>. В ответ в нижней части окна отобразятся команды системы info. Перемещаться в этом разделе позволяют клавиши со стрелками. Чтобы отменить

подсказку, нажмите комбинацию клавиш <Ctrl+X><0>, т.е. сначала нажмите комбинацию клавиш <Ctrl+X>, затем отпустите эти клавиши и нажмите клавишу <0>.

И наконец, чтобы закончить работу с системой info, нажмите клавишу <Q>, и вы вернетесь в оболочку.

Определение путей к исполняемым, исходным файлам и страницам справочного руководства

```
whereis
```

Команда whereis чрезвычайно полезна: она сообщает путь к исполняемому файлу программы, ее исходным файлам (если они существуют) и соответствующим страницам справочного руководства. Например, для kword (текстового процессора из набора Koffice) можно получить следующую информацию (она будет предоставлена при условии, что двоичные, исходные и справочные файлы были установлены):

```
$ whereis kword
kword: /usr/src/koffice-1.4.1/kword /usr/bin/kword
  /usr/bin/X11/kword usr/share/man/man1/kword.1.gz
```

Сначала команда whereis сообщает расположение исходных файлов, /usr/src/koffice-1.4.1/kword, затем информирует о местонахождении исполняемых файлов: /usr/bin/kword и /usr/bin/X11/kword. Как видите, программа kword обнаружена в двух позициях файловой системы. Это несколько необычно, но, тем не менее, в этом нет ничего странного. И наконец, вы получаете информацию о том, где находятся страницы справочного руководства: /usr/share/man/man1/kword.1.gz. Теперь вы знаете, что программа действительно инсталлирована на вашем компьютере, и можете запустить ее.

Если вас интересуют только исполняемые файлы, при вызове whereis надо указать опцию -b.

```
$ whereis -b kword
kword: /usr/bin/kword /usr/bin/X11/kword
```

Опция -m нужна тем, кому необходимы лишь страницы справочного руководства.

```
$ whereis -m kword
kword: /usr/share/man/man1/kword.1.gz
```

И наконец, если вы хотите получать только сведения об исходных файлах, укажите при вызове whereis опцию -s.

```
$ whereis -s kword
kword: /usr/src/koffice-1.4.1/kword
```

Команда whereis позволяет быстро получить чрезвычайно важную информацию о программах. Со временем вы начнете ее использовать гораздо чаще, чем предполагаете сейчас.

Описание команд

```
what is
```

Ранее в этой главе мы рассмотрели опцию -f команды man, которая выводит на экран описание команды, содержащееся на странице справочного руководства. Если вы не забудете опцию -f, вы всегда сможете получить требуемую информацию. Но возможно, вам легче будет запомнить команду whatis. которая делает в точности то же самое — отображает описание команды.

```
$ man -f ls
ls (1) - list directory contents
$ whatis ls
ls (1) - list directory contents
```

Кроме того, команда whatis поддерживает регулярные выражения и символы групповых операций. Для того чтобы выполнить поиск в базе с использованием символов групповых операций, надо задать опцию -w (или --wildcard).

```
$ whatis -w ls*
ls (1) - list directory contents
lsb (8) - Linux Standard Base support for Debian
lshal (1) - List devices and their properties
lshw (1) - list hardware
lskat (6) - Lieutnant Skat card game for KDE
[Информация сокращена для экономии места]
```

Использование символов групповых операций несколько замедляет обработку, однако на современных компьютерах задержка практически незаметна.

Для работы с регулярными выражениями предусмотрена опция -r (или --regex).

```
$ whatis -r ^rm.*
rm (1) - remove files or directories
rmail (8) - handle remote mail received via uucp
rmdir (1) - remove empty directories
rmt (8) - remote magtape protocol module
```

Совет

В данной книге регулярные выражения не рассматриваются, однако вы можете ознакомиться с ними по другим источникам. В частности, информация о них содержится в книге Бена Форты *Освой самостоятельно регулярные выражения. 10 минут на урок* (ИД "Вильямс", 2004 г.).

Регулярные выражения могут замедлять работу, но на достаточно быстродействующей машине вы не заметите этого.

Запомнить команду whatis достаточно просто, по крайней мере, проще, чем man -f. Она позволяет быстро найти важную информацию, так что возьмите ее на вооружение.

Поиск информации о команде по выполняемым ею действиям

```
apropos
```

Как вы уже знаете, команда whatis является аналогом man -f. Аналогично, команда apropos действует подобно man -k. Обе команды ищут на страницах справочного руководства имена и описания команд. Использовать их целесообразно тогда, когда вы знаете, что делает та или иная команда, но забыли ее имя.

На заметку

Слово apropos используется нечасто, но оно существует и означает нечто вроде "имеющий отношение" или "уместный". По значению на него похоже слово "appropriate", но оно и слово "apropos" имеют разные латинские корни. Если вы не верите мне, обратитесь к Web-узлу www.answers.com.

Использовать команду apropos несложно, надо лишь задать после имени команды слово или фразу, которые описывают интересующую вас команду.

```
$ man list
No manual entry for list
$ man -k list
last (1) - show listing of last logged in users
ls (1) - list directory contents
lshw (1) - list hardware
lsof (8) - list open files
[Информация сокращена для экономии места]
```

```
$ apropos list
last (1) - show listing of last logged in users
ls (1) - list directory contents
lshw (1) - list hardware
lsof (8) - list open files
[информация сокращена для экономии места]
```

Так же как и в команде `whatis`, вы можете использовать при поиске опции `-w` (или `--wildcard`) и `-r` (или `--regex`). Поддерживается также опция `-e` (или `--exact`); если она указана, требуется точное совпадение слова или фразы. Например, в последнем примере присутствует команда `last`, так как в ее описании есть слово `listing`. Попробуем повторить поиск, но укажем опцию `-e`. Получим следующий результат:

```
$ apropos -e list
ls (1) - list directory contents
lshw (1) - list hardware
lsof (8) - list open files
[информация сокращена для экономии места]
```

На этот раз описание команды `last` не отображается, поскольку нам нужны результаты, в точности соответствующие слову `list`, поэтому слово `listing` не принимается во внимание. Список результатов поиска по слову `list` насчитывает 80 пунктов и сокращается до 55 при указании опции `-e`. В некоторых случаях эта опция позволяет быстрее находить нужные сведения.

Сведения об экземпляре программы для запуска

```
which
```

Вспомните команду `whereis` и результаты ее применения к команде `kword` с опцией `-b`.

```
$ whereis -b kword
kword: /usr/bin/kword /usr/bin/X11/kword
```

В файловой системе есть два исполняемых файла kword. Но какой из них будет запущен при указании имени kword в командной строке? Выяснить это можно, выполнив команду which.

```
$ which kword
/usr/bin/kword
```

Команда which сообщает о том, какой вариант команды будет выполнен, если вы зададите ее имя. Если вы введете слово kword, а затем нажмете клавишу <Enter>, ваша оболочка выполнит тот файл, который находится в каталоге /usr/bin. Если вы захотите выполнить тот файл, который расположен в каталоге /usr/bin/X11, вам надо сделать этот каталог текущим посредством команды cd, а затем ввести ./kword. Можно также указать полный путь к файлу: /usr/bin/X11/kword.

Команда which также представляет собой сверхбыстрый способ выяснить, поддерживается ли та или другая команда в вашей системе. Если команда может быть выполнена и путь к соответствующему каталогу указан в переменной окружения PATH, вы получите информацию о том, где найти ее. В противном случае отобразится лишь приглашение для ввода очередной команды.

```
$ which arglebargle
$
```

Если вы хотите определить местоположение всех экземпляров файлов, поддерживающих команду (как в случае, если вы задаете выражение whereis -b), вам надо использовать опцию -a (первая буква слова "all").

```
$ which -a kword
/usr/bin/kword
/usr/bin/X11/kword
```

Выводы

Название данной главы говорит о том, что в ней будет продолжен разговор о командах. Теперь вы знаете, что существуют самые разнообразные способы получить дополнительную информацию о тех средствах, которые доступны из командной строки. Основными из них являются команды `man` и `info`. Они используют огромные объемы данных, описывающих практически все команды, доступные на компьютере под управлением Linux. Нельзя также забывать команды `whereis`, `whatis`, `apropos` и `which`. Они очень полезны, в особенности тогда, когда вам не хочется вникать в детали использования команд `man` и `info`. Но далеко не всегда они могут предоставить необходимые сведения. Время от времени приходится читать страницы справочного руководства. Это так же, как и с овощами. Не все любят их, но они полезны.

Многие команды, описанные в данной главе, частично дублируют друг друга. Например, `man -k` — это то же самое, что `apropos`, `man -f` можно заменить `whatis`, а `whereis -b` функционально эквивалентна `which -a`. За вами выбор, какую из них использовать в каждой конкретной ситуации. Несмотря на то что разные команды выполняют одни и те же действия, желательно знать их все. Это поможет вам читать сценарии, написанные другими. Система Linux отличается тем, что предоставляет пользователю богатый выбор возможностей, даже тогда, когда это касается таких простых вещей, как команды, запускаемые посредством оболочки.

4

Объединение команд

В детстве мы все учили числа, а затем учились выполнять с ними арифметические действия, используя для записи символы +, -, × и =. На данный момент мы знаем несколько команд, но выполнять их умеем только по одной. На самом деле команды можно объединять в сложные и интересные конструкции посредством различных операций. Для обозначения этих операций используются символы |, >, >>, < и др.

В этой главе мы рассмотрим "строительные блоки", которые позволят гораздо эффективнее использовать как знакомые вам команды, так и те, которые будут подробно обсуждаться в последующих главах.

Последовательное выполнение нескольких команд

;

Предположим, что вам надо выполнить одну за другой несколько команд и среди них есть такие, которые выполняются достаточно долго. Значит ли это, что вам придется неотлучно сидеть возле компьютера? Допустим, у вас есть большое число записей Джона Колтрейна в формате mp3, которые представлены в виде zip-архива. Вы хотите

распаковать их, поместить в новый подкаталог, а затем удалить архив. Пока что вы умеете лишь выполнять команды по одной, например:

```
$ ls -l /home/scott/music
-rw-r--r--  1437931  2005-11-07 17:19
  JohnColtrane.zip
$ unzip /home/scott/music/JohnColtrane.zip
$ mkdir -p /home/scott/music/coltrane
$ mv /home/scott/music/JohnColtrane*.mp3
  /home/scott/music/coltrane/
$ rm /home/scott/music/JohnColtrane.zip
```

На заметку

Для экономии места имена владельца и группы удалены.

Размер файла `John_Coltrane.zip` равен 1,4 Гбайт. Даже на быстродействующей машине для его распаковки потребуется длительное время, в течение которого у вас наверняка найдется более интересное занятие, чем сидеть возле компьютера, ожидая завершения операции. Решение проблемы — в объединении команд.

Для объединения команд достаточно записать их все в одной строке, отделив друг от друга точкой с запятой. При этом команды будут выполняться последовательно одна за другой. Выполнение очередной команды начнется лишь после завершения (неважно, успешного или нет) предыдущей.

Объединенные команды записываются следующим образом:

```
$ ls -l /home/scott/music
-rw-r--r-- 1437931 2005-11-07 17:19 JohnColtrane.
zip
$ unzip /home/scott/music/JohnColtrane.zip ;
```

```
mkdir -p /home/scott/music/coltrane ;
mv /home/scott/music/JohnColtrane*.mp3
/home/scott/music/coltrane/ ;
rm /home/scott/music/JohnColtrane.zip
```

Каждая команда выполняется, затем завершается, и управление передается следующей команде. Записать несколько команд подобным образом достаточно просто, и такой подход реально сэкономит вам время.

Перед выполнением команды можно задавать небольшую задержку. Если вы хотите сохранить копию экрана, вам поможет следующая объединенная команда (для ее выполнения надо, чтобы на вашем компьютере был установлен продукт ImageMagick; обычно он входит в состав дистрибутивного пакета Linux):

```
$ sleep 3 ; import -frame window.tif
```

В данном случае команда `sleep` реализует задержку на три секунды, затем посредством команды `import` обеспечивается копирование содержимого экрана. За эти три секунды вы можете переместить требуемое окно на передний план, минимизировать другие окна и выполнить прочие несложные действия. Символ ; очень удобен для логического разделения команд, поэтому он используется достаточно часто.

Внимание!

Объединяя команды, будьте внимательны, особенно, если вы удаляете или перемещаете файлы. Убедитесь в том, что вы правильно ввели команды, в противном случае результаты могут быть непредсказуемыми.

Выполнение команды при условии успешного завершения предыдущих

&&

Из предыдущего раздела вы узнали, что символ ; может служить разделителем при объединении команд, например:

```
$ unzip /home/scott/music/JohnColtrane.zip ;
  mkdir -p /home/scott/music/coltrane ;
  mv /home/scott/music/JohnColtrane*.*mp3
  /home/scott/music/coltrane/ ;
  rm /home/scott/music/JohnColtrane.zip
```

Но предположим, что вы допустили ошибку и ввели следующее:

```
$ unzip /home/scott/JohnColtrane.zip ;
  mkdir -p /home/scott/music/coltrane ;
  mv /home/scott/music/JohnColtrane*.*mp3
  /home/scott/music/coltrane/ ;
  rm /home/scott/music/JohnColtrane.zip
```

Вместо `unzip /home/scott/music/John_Coltrane.zip` было случайно задано `unzip /home/scott/John_Coltrane.zip`. Вы не заметили ошибки, нажали клавишу <Enter> и ожидаете окончания выполнения последовательности команд. Однако команда `unzip /home/scott/John_Coltrane.zip` не может быть выполнена, поскольку файл не существует. Тем не менее последовательность команд продолжается как ни в чем не бывало, и управление передается команде `mkdir`, которая завершается успешно. Третья команда (`mv`) также не может быть выполнена, так как файлы mp3, которые требуется переместить, не существуют; они не были созданы командой `unzip`. И наконец, выполняется четвертая команда, удаляющая zip-файл

(заметьте, что на этот раз вы задали правильный путь). Теперь вам остается лишь восстанавливать файл из резервной копии (если она существует) и начинать все сначала.

На заметку

Вы не верите, что описанное возможно? Я сам совершил нечто подобное несколько дней назад. Ощущение было довольно скверным.

Источник проблемы — символ ;, который вызывает последовательное выполнение команд без учета того, успешно ли они завершаются. Гораздо лучшее решение — разделять команды символами &&. В этом случае они также выполняются последовательно, но следующая команда получает управление лишь в том случае, если предыдущая завершилась без ошибки (т.е. если она возвращает код завершения, равный 0). В случае ошибки выполнение всей цепочки команд прекращается. Если вы укажете вместо ; символы &&, выражение в командной строке примет следующий вид:

```
$ unzip /home/scott/JohnColtrane.zip &&
  mkdir -p /home/scott/music/coltrane &&
  mv /home/scott/music/JohnColtrane.*mp3
  /home/scott/music/coltrane/ &&
  rm /home/scott/music/JohnColtrane.zip
```

Поскольку первая команда `unzip` не может успешно завершиться, прекращается весь процесс. Когда вы обнаружите это, файл `John_Coltrane.zip` будет по-прежнему находиться на диске и вы сможете начать все сначала. Это гораздо лучше, чем остаться без архивного файла, не так ли?

Рассмотрим два примера, демонстрирующих преимущества операции &&. В главе 13 вы ознакомитесь с инструментом `apt`, существенно упрощающим обновление версии Debian системы Linux. При использовании `apt` сначала обновляется список доступного программного обеспечения,

а затем выясняется, есть ли соответствующие дополнения. Если список программ не обновлен, вам не надо даже искать дополнения. Для того чтобы без необходимости не выполнять второй процесс (поиск дополнений), надо разделять команды посредством символов &&.

```
# apt-get update && apt-get upgrade
```

Второй пример выглядит так. Предположим, вы хотите преобразовать PostScript-файл в файл PDF, используя команду `ps2pdf`, затем вывести PDF-файл на печать и удалить файл PostScript. Здесь также пригодится разделитель &&.

```
$ ps2pdf foobar.ps && lpr foobar.pdf &&
  rm foobar.ps
```

Если вместо && вы укажете ;, а `ps2pdf` не выполнит свою задачу, файл PostScript будет удален, и вы даже не сможете повторить попытку из-за отсутствия исходных данных.

Вы убедились в том, что в ряде случаев предпочтительнее использовать разделитель &&? Если нет опасности удалить файл, вполне может подойти разделитель ;, но если в наборе команд присутствует `rm` или другая команда, подобная ей, лучше использовать для разделения символы &&.

Выполнение команды при условии, что предыдущая завершилась с ошибкой

```
||
```

Разделитель && указывает на то, что очередная команда должна быть выполнена только при условии успешного завершения предыдущей. Символы || действуют противоположным образом. Если первая команда завершается с ошибкой (т.е. если она возвращает состояние завершения, отличное от 0), то только в этом случае управление передается следующей команде. Подобную конструкцию можно

рассматривать как выражение типа "или, или", т.е. выполняется либо первая команда, либо вторая.

Разделители || часто используются для оповещения администратора об остановке процесса. Например, чтобы убедиться, что определенный компьютер работает нормально, его надо постоянно опрашивать посредством команды ping (подробнее эта команда будет рассматриваться в главе 14). Если ping сообщит об ошибке, администратору будет отправлено почтовое сообщение.

```
ping -c 1 -w 15 -n 72.14.203.104 ||
{
    echo "Server down"  | mail -s
      'Server down' admin@google.com
}
```

Немного поразмыслив, вы найдете множество применений разделителя ||. Он давно взят на вооружение многими администраторами.

Использование выходных данных одной команды при вызове другой команды

```
$()
```

Механизм подстановки команд позволяет включить выходные данные одной команды в другую команду. При этом вторая команда будет выполняться так, как будто вы ввели все символы вручную. Первую команду — ту, выходные данные которой надо включить во вторую команду, — следует поместить в круглые скобки и поставить перед открывающей скобкой символ $. Использование подстановки команд пояснит вам приведенный ниже пример.

Предположим, вы пришли домой с банкета, подключили цифровую камеру к компьютеру, чтобы извлечь из нее

новые фотоснимки и поместить их в каталог, имя которого соответствует текущей дате.

```
$ pwd
/home/scott/photos/family
$      ls -1F
2005-11-01/
2005-11-09/
2005-11-15/
$ date "+%Y-%m-%d"
2005-11-24
$ mkdir $(date "+%Y-%m-%d")
$      ls -1F
2005-11-01/
2005-11-09/
2005-11-15/
2005-11-24/
```

В данном примере первой выполнится команда `date "+%Y-%m-%d"`, а затем ее выходные данные (2005-11-24) будут использованы командой `mkdir` для формирования имени нового каталога. Подстановка команд — очень мощный механизм, и если вы просмотрите сценарии, написанные другими специалистами (их можно легко найти в Web), то обнаружите, что подстановка используется в них очень часто.

На заметку

Раньше для организации подстановки применялся символ ` (клавиша в верхней левой части клавиатуры). Теперь же рекомендуется применять более привычные символы — `$()`.

Входной и выходной потоки

Для того чтобы понять материал, изложенный в данной главе, необходимо знать, что оболочка Linux поддерживает

три потока: стандартный входной поток (или стандартный ввод), стандартный выходной поток (или стандартный вывод) и стандартный поток ошибок. Каждому из этих потоков поставлен в соответствие дескриптор файла (числовой идентификатор) и общепринятое сокращение, кроме того, тот или иной поток может использоваться по умолчанию.

Например, если вы вводите информацию с клавиатуры, вы передаете данные в стандартный входной поток, которому соответствует дескриптор 0 и сокращение `stdin`. Когда ваш компьютер выводит данные на терминал, он использует стандартный выходной поток, которому соответствует дескриптор 1 и сокращение `stdout`. И наконец, если система должна сообщить об ошибке, используется стандартный поток ошибок, которому соответствует дескриптор 2 и сокращение `stderr`.

Обсудим работу с потоками на примере известной вам команды `ls`. Когда вы вводите информацию с клавиатуры, вы используете `stdin`. После того как вы ввели `ls` и нажали клавишу <Enter>, список файлов и каталогов выводится в `stdout`. Если вы зададите в качестве параметра данной команды несуществующий каталог, сообщение об ошибке будет выведено на ваш монитор посредством `stderr`. В табл. 4.1 поясняется использование потоков.

Таблица 4.1. Потоки ввода-вывода

Дескриптор файла (идентификатор)	Название	Общепринятое сокращение	Использование по умолчанию
0	Стандартный входной поток	`stdin`	Клавиатура
1	Стандартный выходной поток	`stdout`	Терминал
2	Стандартный поток ошибок	`stderr`	Терминал

В этой главе вы научитесь перенаправлять ввод и вывод. Вместо того чтобы выводить выходные данные на терминал, вы сможете, например, передать их другой программе. Точно так же, вместо того, чтобы вводить данные с клавиатуры, можно организовать получение их из файла. После того как вы поймете принципы использования `stdin` и `stdout`, вы сможете проделывать с ними поистине головокружительные трюки.

Передача выходных данных одной команды на вход другой команды

```
|
```

Всем известно, что система Unix собрана из отдельных частей, слабо зависящих друг от друга. Ничто не иллюстрирует этот принцип так ярко, как механизм каналов. Канал задается символом `|`, который помещается между двумя командами. Канал организует передачу выходных данных первой команды на вход второй команды. Иными словами, `|` перенаправляет `stdout` так, что он соединяется с потоком `stdin` следующей команды.

Ниже приведен простой пример, позволяющий лучше понять механизм каналов. Вы уже знаете команду `ls`, а в главе 5 ознакомитесь с командой `less`, позволяющей постранично просматривать текст на экране. Если вы примените команду `ls` к каталогу, содержащему множество файлов, например `/usr/bin`, то выходные данные быстро промелькнут на экране и их невозможно будет прочитать. Если же посредством канала вы соедините выход `ls` с входом `less`, то информация будет отображаться постранично.

```
$ pwd
/usr/bin
$ ls -1
zipinfo
zipnote
zipsplit
zsoelim
zxpdf
[Список файлов сокращен.
 Полностью он занимает 2318 строк!]
$ ls -1 | less
411toppm
7z
7za
822-date
a2p
```

Связывание команды `ls -1` с командой `less` посредством канала упростит вашу работу.

Рассмотрим более сложный пример; в нем использованы две команды `ps` и `grep`, которые будут обсуждаться позже (прочитав главу 9, вы узнаете, что команда `ps` предоставляет информацию о выполняющихся процессах, а в главе 12 будет сказано, что `grep` предназначена для поиска строк в файлах). Предположим, что программа Firefox начала вести себя странным образом, и вы подозреваете, что это происходит потому, что в фоновом режиме выполняется большое число ее экземпляров. Команда `ps` сообщает о каждом процессе на вашем компьютере, но в большинстве случаев объем выходных данных очень большой. Связав выход `ps` с входом `grep` и выполнив поиск по слову `firefox`, вы сможете немедленно выяснить, выполняется ли данная программа и сколько экземпляров ее присутствует в системе.

```
$ ps ux
1504  0.8  4.4  75164 46124 ? S Nov20 1:19
  kontact
19003  0.0  0.1   3376  1812 pts/4 S+ 00:02
  0:00 ssh admin@david.hartley.com
21176  0.0  0.0      0     0 ? Z 00:14 0:00
  [wine-preloader] <defunct>
24953  0.4  3.3  51856 34140 ? S 00:33 0:08
  kdeinit: kword
  /home/scott/documents/clientele/current
[Информация сокращена для экономии места]
$ ps ux | grep firefox
scott  8272  4.7 10.9 184072 112704 ? Sl Nov19
  76:45 /opt/firefox/firefox-bin
```

Из 58 строк выходных данных осталась одна — именно в ней и содержится интересующая вас информация.

На заметку

Учтите, что не все программы могут работать с каналами. Например, текстовый редактор `vim` (или `pico`, или `nano`, или `emacs`) забирает у оболочки управление так, что данные с клавиатуры непосредственно передаются `vim`, а выходная информация отображается средствами самой программы. Поскольку `vim` полностью контролирует оболочку, вы не можете использовать каналы для перенаправления вывода. Со временем, поработав с оболочкой, вы научитесь распознавать программы, к которым не применим механизм каналов.

Перенаправление выходных данных в файл

```
>
```

В обычных условиях данные, сгенерированные в результате выполнения команды, выводятся на экран или в

поток stdout. Если вы хотите, чтобы информация выводилась не на экран, а в файл, вы можете сделать это, указав символ >.

```
$ pwd
/home/scott/music
$      ls -1F
Hank_Mobley/
Horace_Silver/
John_Coltrane/
$ ls -1F Hank_Mobley/* > hank_mobley.txt
$ cat hank_mobley.txt
1958_Peckin' _Time/
1960_Roll_Call/
1960_Soul_Station/
1961_Workout/
1963_No_Room_For_Squares/
$      ls -1F
Hank_Mobley/
hank_mobley.txt
Horace_Silver/
John_Coltrane/
```

Заметьте, что когда вы вызвали команду `ls -F` в первый раз, файл `hank_mobley.txt` отсутствовал. Он был создан, когда вы использовали символ > для перенаправления вывода. Поступая подобным образом, необходимо соблюдать осторожность: если бы файл `hank_mobley.txt` существовал ранее, он был бы полностью заменен.

Внимание!

И еще раз напоминаю: соблюдайте осторожность, используя перенаправление! Вы можете разрушить файл с важными данными.

Как предотвратить перезапись файла при перенаправлении

Существует способ предотвратить запись информации в файл поверх существующей при перенаправлении вывода. Если вы установите опцию noclobber, оболочка bash выполнит перенаправление в существующий файл только с вашего разрешения. Чтобы установить noclobber, надо выполнить следующую команду:

```
$ set -o noclobber
```

Теперь, если вы захотите перезаписать файл путем перенаправления вывода, вместо > вам придется указать >|, как показано в следующем примере:

```
$ pwd
/home/scott/music
$ ls -1F
Hank_Mobley/
hank_mobley.txt
Horace_Silver/
John_Coltrane/
$ ls -1F Hank_Mobley/* > hank_mobley.txt
ERROR
$ ls -1F Hank_Mobley/* >| hank_mobley.txt
$ cat hank_mobley.txt
1958_Peckin' _Time/
1960_Roll_Call/
1960_Soul_Station/
1961_Workout/
1963_No_Room_For_Squares/
```

Сбросить noclobber можно таким образом:

```
$ set +o noclobber
```

Чтобы опция `noclobber` была установлена постоянно, вам надо включить в файл `.bashrc` выражение `set -o noclobber`.

Перенаправление выходных данных и запись их в конец файла

```
>>
```

Как вы уже знаете, символ > задает перенаправление выходных данных из `stdout` в файл. Например, можно легко перенаправить в файл вывод команды, возвращающей информацию о дате.

```
$ date
Mon Nov 21 21:33:58 CST 2005
$ date > hank_mobley.txt
$ cat hank_mobley.txt
Mon Nov 21 21:33:58 CST 2005
```

Однако нельзя забывать, что если указанный файл отсутствует, он будет создан при выводе перенаправленной информации; если же файл существует, его содержимое будет полностью заменено. Указав >> вместо >, вы присоедините выходные данные в конец имеющегося файла (если файл отсутствует, он, как и раньше, будет создан).

```
$ cat hank_mobley.txt
Mon Nov 21 21:33:58 CST 2005
$ ls -1F Hank_Mobley/* >> hank_mobley.txt
$ cat hank_mobley.txt
Mon Nov 21 21:33:58 CST 2005
1958_Peckin' _Time/
1960_Roll_Call/
1960_Soul_Station/
1961_Workout/
1963_No_Room_For_Squares/
```

Внимание!

Перенаправляя ввод, удостоверьтесь, что вы указали именно символы >>. Если вы случайно зададите >, вместо записи в конец файла будет полностью заменено его содержимое.

Использование содержимого файла в качестве входных данных

<

В обычных условиях выходные данные задаются с клавиатуры, т.е. читаются из stdin. Подобно тому как stdout можно перенаправить в файл, вы можете перенаправить stdin так, что входные данные будут поступать не с клавиатуры, а из файла. Зачем это нужно? Некоторые программы не могут самостоятельно открывать файлы. В этом случае решить проблему позволяет символ <, перенаправляющий ввод.

Рассмотрим в качестве примера команду echo, которая повторяет данные, введенные через stdin.

```
$ echo "This will be repeated"
This will be repeated.
```

Однако вы можете перенаправить входную информацию посредством символа <, и команда echo обработает вместо символов, введенных с клавиатуры, содержимое файла. Ниже посредством команды echo выводится содержимое файла hank_mobley.txt, созданного в предыдущем разделе.

```
$ echo < hank_mobley.txt
Mon Nov 21 21:33:58 CST 2005
1958_Peckin' _Time/
1960_Roll_Call/
1960_Soul_Station/
1961_Workout/
1963_No_Room_For_Squares/
```

Естественно, символ < не используется постоянно, однако бывают ситуации, когда он приходится как нельзя кстати.

Выводы

В начале этой книги мы рассмотрели несколько простых команд, а сейчас вы ознакомились со "строительными блоками", которые позволяют объединять эти команды. Материал последующих глав будет более сложным; в них вы узнаете об обширных возможностях, которые предоставляет Linux. Однако воспользоваться ими можно лишь, владея средствами, рассмотренными в данной главе.

5

Отображение содержимого файлов

Одна из особенностей системы Linux состоит в том, что практически все конфигурационные файлы, файлы протоколов и файлы с информацией о системе представлены в формате ASCII. Поскольку такой подход известен уже очень давно (по меркам истории компьютерных систем), существует множество команд, предназначенных для просмотра содержимого текстовых файлов.

В этой главе мы рассмотрим четыре команды, наиболее часто применяемые для чтения ASCII-данных.

Вывод содержимого файла в stdout

```
cat
```

Пользователям системы DOS доступна команда `type`, отображающая содержимое текстового файла на экране. Пользователи Linux могут применять для той же цели команду `cat`.

```
$ cat Shakespeare_-_Hamlet_-_fragment.txt
Как к чудесам, вы к ним и отнеситесь.
Гораций, много в мире есть того,
Что вашей философии не снилось.
$
```

Команда cat выводит файл на экран, а затем возвращает управление оболочке. Если содержимое файла не помещается на экране, вам надо воспользоваться средствами прокрутки.

При использовании команды cat может возникать следующая проблема: если объем документа, предназначенного для просмотра, слишком велик, текст быстро промелькнет на экране, и прочитать его будет невозможно (попробуйте для примера выполнить команду cat Melville_-_Moby_Dick.txt; конечно, соответствующий файл должен присутствовать в вашей файловой системе). Выход — использовать команду less, которая будет обсуждаться далее в этой главе. Она обеспечивает постраничный вывод текста.

Конкатенация файлов и вывод их в stdout

```
cat файл_1 файл_2
```

Имя команды cat представляет собой сокращение от слова "concatenate" (конкатенация), т.е. объединение файлов. Основное назначение данной команды — объединение нескольких файлов в один. Вывод на экран одного файла — лишь частный случай ее использования. Предположим, что у вас есть фрагмент сонета Шекспира и фрагмент текста Брэдбери, которые вы хотите просмотреть одновременно.

```
$ cat Shakespeare_-_sonnet_121_fragment.txt
  Bradbury_-_Mars_fragment.txt
Уж лучше быть, чем только слыть дурным,
Упрекам подвергаться понапрасну.
Ведь даже радость превратится в дым,
Когда не сам признал её прекрасной.
Рэй Бредбери. Марсианские хроники
```

```
РАКЕТНОЕ ЛЕТО
  Только что была огайская зима: двери заперты,
окна закрыты, стекла незрячие от изморози, все
крыши оторочены сосульками, дети мчатся с горок на
лыжах, женщины в шубах черными медведицами бредут
по гололедным улицам.
```

Заметьте, что команда `cat` не включает между файлами ни строку дефисов, ни другой разделитель. Тексты, содержащиеся в файлах, следуют непосредственно один за другим. Если вы хотите, чтобы они были разделены, надо предусмотреть соответствующие данные в исходных файлах, например, включить в конец первого файла пустую строку.

Конкатенация файлов и запись результатов в другой файл

```
cat файл_1 файл_2 > файл_3
```

В предыдущем примере мы объединяли два файла и отображали их на экране, т.е. выводили в `stdout`. Однако такое решение не всегда устроит вас. В ряде случаев, обрабатывая командой `cat` два файла, надо сохранить результаты и в файле. Сделать это можно, перенаправив вывод в файл (см. главу 4).

```
$ ls
housman_-_rue.txt   quarles_-_the_world.txt
$ cat housman_-_rue.txt
  quarles_-_the_world.txt > poems.txt
$ ls
housman_-_rue.txt          poems.txt
quarles_-_the_world.txt
```

Теперь мы можем выполнять с файлом poems.txt. любые действия. Например, можно добавить в конец этого файла любой текст.

```
$ cat housman_-_one-and-twenty.txt >> poems.txt
```

Заметьте, что на этот раз мы использовали символы >>. Это существенно, потому что, например, следующая команда не дала бы желаемых результатов:

```
$ cat housman_-_one-and-twenty.txt poems.txt >
  poems.txt
```

Если вы попытаетесь выполнить конкатенацию содержимого файла и записать результаты в этот же файл, то получите сообщение об ошибке.

```
cat: poems.txt: input file is output file
```

Конкатенация файлов и нумерация строк

```
cat -n файл_1 файл_2
```

При работе с текстом, а в особенности с исходным кодом программы, удобно, чтобы строки были пронумерованы. Для того чтобы сгенерировать номера строк, надо использовать при вызове cat опцию -n (или --number).

```
cat -n Shakespeare_-_sonnet_121_fragment.txt
  Spenser_-_sonnet_fragment.txt
     1   Уж лучше быть, чем только слыть дурным,
     2   Упрёкам подвергаться понапрасну.
     3   Ведь даже радость превратится в дым,
     4   Когда не сам признал её прекрасной.
     5   В безбрежном океане звёздный луч
     6   Поможет к гавани корабль вести,
     7   Но развернётся полог чёрных туч,
     8   И мореход сбивается с пути.
```

В данном примере объединены фрагменты сонетов Шекспира и Спенсера, и их строки пронумерованы. Таким образом, команда `cat` позволят без труда получить полезную дополнительную информацию.

На заметку

Если вам нужны дополнительные функциональные возможности, попробуйте вместо `cat` использовать команду `dog`. С ее помощью можно просматривать не только локальные файлы, но и исходный HTML-код Web-страниц и даже отображать списки изображений или ссылок, встречающихся на странице. Команда `dog` может преобразовывать символы нижнего регистра в верхний регистр и, наоборот, поддерживает окончания строк в форматах Mac OS, DOS и Unix, а также позволяет выводить заданный фрагмент текста (например, строки 5–25). К тому же страница справочного руководства, посвященная команде `dog`, написана очень хорошо.

По аналогии с командой `cat` создана еще одна команда — `tac`. Как вы, вероятно, заметили, имя `tac` — это `cat` наоборот. Команда `tac` объединяет файлы в обратном порядке. Потребность в ней возникает достаточно редко, но полезно знать, что такая возможность существует.

Постраничный вывод текста

```
less файл
```

В ряде случаев команда `cat` очень полезна, но она совершенно не подходит для просмотра файлов большого объема, так как при этом по экрану пробегает сплошной поток символов, прочитать которые практически невозможно. Если вам надо просмотреть длинный файл (в данном случае длинным можно считать файл, содержимое которого не помещается на экране), вместо команды `cat` лучше использовать команду `less`.

Команда `less` организует постраничный вывод текста. Подобные возможности обеспечивают команды `more` и `pg`. Сама команда `less` была реализована в 1985 г. как расширение команды `more`.

С помощью `less` файл открывается предельно просто.

```
$ less Paradise_Lost.txt
```

Текст, выводимый командой `less`, занимает весь экран; для навигации используется клавиатура. При желании вы можете завершить выполнение `less` и вернуться в командную строку. Для управления отображением текста используются клавиши, описанные в табл. 5.1.

Таблица 5.1. Основные клавиши команды `less`

Основные клавиши	Выполняемые действия
<PageDn>, e или пробел	Перемещение вперед на одну страницу
<PageUp> или b	Перемещение назад на одну страницу
<Enter>, e, j или стрелка вниз	Перемещение вперед на одну строку
y, k или стрелка вверх	Перемещение назад на одну строку
G или p	Перемещение вперед к концу файла
1G	Перемещение назад к началу файла
<Esc+)> или стрелка вправо	Прокрутка вправо
<Esc+(> или стрелка влево	Прокрутка влево
Q	Завершение работы команды `less`

Как видите, одни и те же действия можно выполнить различными способами. Чаще всего при работе с `less` используются клавиши, сдвигающие текст вперед на одну страницу и завершающие работу программы.

Для отображения информации о файле надо нажать клавишу <=>. В результате в нижней части экрана будет выведена информация, подобная представленной ниже.

```
Paradise_Lost.txt lines 7521-7560/10762 byte
  166743/237306 70% (press RETURN)
```

Для того чтобы вернуться в режим просмотра текста, надо нажать клавишу <Enter>.

Нумерацию строк можно реализовать не только средствами `cat`, но также с помощью команды `less`. Очевидно, что номера строк будут присутствовать лишь во время работы с командой `less`. После нажатия клавиши <Q> номера исчезнут. Для того чтобы в начале каждой строки отображался ее номер, надо при вызове `less` указать опцию `-N` (или `--LINE-NUMBERS`).

```
$ less -N Paradise_Lost.txt
```

Поиск с помощью программы постраничного просмотра

Если вы просматриваете с помощью `less` большой файл, бывает трудно обнаружить нужный фрагмент текста. Предположим, например, что вас интересует, называл ли Джон Мильтон в "Потерянном рае" яблоком плод, ставший причиной изгнания Адама и Евы. Работая с программой `less`, введите символ / и укажите шаблон для поиска. При необходимости вы можете даже использовать регулярные выражения. Окончив ввод шаблона, нажмите клавишу <Enter> и программа `less` представит вам первый фрагмент текста, соответствующий шаблону (если такой фрагмент существует). Если поиск закончится неудачей, вы получите следующее сообщение.

```
Pattern not found (press RETURN)
```

При необходимости вы можете повторить поиск как вперед, так и назад по тексту. В табл. 5.2 описаны средства поиска, поддерживаемые командой `less`.

Таблица 5.2. Команды поиска, предусмотренные в программе `less`

Команда	Действие
/шаблон	Поиск в прямом направлении (возможно использование регулярных выражений)
n	Повторный поиск в прямом направлении
N	Повторный поиск в обратном направлении

На заметку

Мильтон не упоминал яблоко; он называл его плодом. Я это хорошо знаю, так как долгое время изучал английскую литературу XVII столетия. Я больше не занимаюсь этим. Свидетельством тому — данная книга.

Редактирование файлов, отображаемых средствами постраничного просмотра

Программа `less` — это не редактор, а лишь средство просмотра, однако вы можете передать файл, отображаемый с помощью `less`, текстовому редактору, например `vim` или `nano`. Для редактирования надо нажать клавишу <V>. Попробуйте сделать это. Откройте файл с помощью программы `less` и нажмите клавишу <V>. Через одну-две секунды на экране отобразятся интерфейсные элементы текстового редактора. Внесите необходимые изменения, завершите работу редактора, и вы снова вернетесь в программу `less`, но в ней уже будет отображаться измененный файл.

Если вам не подходит редактор, вызываемый после нажатия клавиши <V>, вы можете заменить его любым другим. Например, если вы хотите использовать `vim`, то перед вызовом `less` надо выполнить следующую команду:

```
$ export EDITOR=vim
```

В течение сеанса работы данную команду достаточно вызвать один раз. После этого всякий раз, когда вы обратитесь к `less`, с ней будет связан редактор `vim`. Если вы начнете новый сеанс, вам снова придется устанавливать редактор. Для того чтобы не делать этого каждый раз вручную, надо включить в файл `.bashrc` следующую строку:

```
export EDITOR=vim
```

Просмотр первых десяти строк файла

`head`

Если вы хотите просмотреть лишь первые десять строк файла, вам не обязательно использовать команду `cat` или `less`. Достаточно вызвать команду `head`, которая выведет первые десять строк файла и завершит свою работу.

```
$ head Carroll_Jabberwocky.txt
Бармаглот

Варкалось. Хливкие шорьки
Пырялись по наве,
И хрюкотали зелюки,
Как мюмзики в мове.
О бойся Бармаглота, сын!
Он так свирлеп и дик,
А в глуще рымит исполин -
Злопастный Брандашмыг.
```

Команда head очень удобна в том случае, если вам надо быстро оценить начало файла, независимо от его размера. Практически мгновенно вы узнаете, тот ли текст содержится в файле, который вы ищете.

Просмотр первых десяти строк нескольких файлов

```
head файл_1 файл_2
```

Команду head можно использовать для просмотра начальных строк нескольких файлов. Полученные при этом данные похожи на те, которые предоставляются командой cat, за исключением того, что отображаемое содержимое файла ограничивается несколькими строками, а между файлами включаются разделители.

```
$ head Paradise_Lost.txt Paradise_Regained.txt
==> Paradise_Lost.txt <==
Джон Мильтон. Потерянный рай
КНИГА ПЕРВАЯ

О первом преслушанье, о плоде
Запретном, пагубном, что смерть принес
И все невзгоды наши в этот мир,
Людей лишил  Эдема, до поры,
Когда нас Величайший Человек
Восставил, Рай блаженный нам вернул,-
Пой, Муза горняя! Сойди с вершин
==> Paradise_Regained.txt <==
Джон Мильтон. Возвращенный рай
КНИГА ПЕРВАЯ

Я пел доселе, как утратил Рай
Преслушный человек; а днесь пою,
```

```
Как Рай людскому роду возвратил
Престойкий Человек, что всяк соблазн
Отверг и, Соблазнителя презрев
Лукавого, осилил и попрал;
И в пустошах воздвигся вновь Эдем.
```

В качестве разделителей используются пустые строки и специальные заголовки. Они позволяют легко отличить один файл от другого.

Просмотр произвольного числа строк из файлов

Если вас не устраивает тот факт, что из каждого файла извлекается ровно десять строк, вы можете указать команде head отображать другой объем текста. Для этой цели используется опция -n, за которой следует число, например 5 (можно также задать опцию --lines=5). Если вы зададите два или более файлов, из начала каждого файла будет выбрано указанное число строк.

```
$ head -т 5 Paradise_Lost.txt
  Paradise_Regained.txt
==> Paradise_Lost.txt <==
Джон Мильтон. Потерянный рай
КНИГА ПЕРВАЯ

О первом преслушанье, о плоде
Запретном, пагубном, что смерть принес
==> Paradise_Regained.txt <==
Джон Мильтон. Возвращенный рай
КНИГА ПЕРВАЯ

Я пел доселе, как утратил Рай
Преслушный человек; а днесь пою,
```

Заметьте, что в число "пяти строк" входят пустые строки, присутствующие в файле.

Просмотр указанного числа байтов из начала файла

```
head -c
```

Опция -n позволяет задать число строк, просматриваемых из начала файла. Но что делать, если вам надо задать не количество строк, а число байтов, или килобайтов, или даже мегабайтов? (Последнее в большинстве случаев бессмысленно, так как объем информации окажется слишком велик.) Для этой цели используется опция -c (или --bytes=).

Например, для того, чтобы просмотреть первые 100 байтов файла, содержащего текст Брэдбери, надо воспользоваться следующей командой:

```
$ head -c 100 Bradbury_-_Mars.txt
Рэй Бредбери. Марсианские хроники
РАКЕТНОЕ ЛЕТО
  Только что была огайская зима: двери заперты, ок
```

Указанное число байтов совсем не обязательно должно прийтись на конец строки или пробел между словами. Так, в данном примере текст обрывается на середине слова.

Для того чтобы отобразить первые 100 Кбайт этого же файла, надо переписать команду таким образом:

```
$ head -c 100k Bradbury_-_Mars.txt
```

Аналогично можно задать отображение первых 100 мегабайтов, но объем всей книги значительно меньше. Формально соответствующая команда имеет следующий вид:

```
$ head -c 100m Bradbury_-_Mars.txt
```

Заметьте, что мегабайт — это 1048576 байтов (1024 × 1024).

Просмотр последних десяти строк файла

```
tail
```

Команда head отображает начало файла. Нетрудно догадаться, что команда `tail` позволяет просмотреть конец файла.

```
$ tail Carroll_Jabberwocky.txt
Ува! Ува! И голова
Барабардает с плеч.
О светозарный мальчик мой!
Ты победил в бою!
О храброславленный герой.
Хвалу тебе пою!
Варкалось. Хливкие шорьки
Пырялись по наве,
И хрюкотали зелюки,
Как мюмзики в мове.
```

Зачем нужна эта команда? Чаще всего она применяется для просмотра окончания файла протокола. Именно там располагаются последние записи.

Просмотр последних десяти строк нескольких файлов

```
tail файл_1 файл_2
```

Если команда head позволяет просматривать первые строки нескольких файлов, логично ожидать, что команда `tail` также может обрабатывать больше одного файла.

```
$ tail Paradise_Lost.txt Paradise_Regained.txt
==> Paradise_Lost.txt <==
Струясь, клубился, а в проеме Врат
Виднелись лики грозные, страша
Оружьем огненным. Они невольно
Всплакнули - не надолго. Целый мир
Лежал пред ними, где жилье избрать
Им предстояло. Промыслом Творца
Ведомые, шагая тяжело,
Как странники, они рука в руке,
Эдем пересекая, побрели
Пустынною дорогою своей.
==> Paradise_Regained.txt <==
До срока вас на муки Он не вверг.
Хвала Господню Сыну, всех миров
Наследнику! Смиритель Сатаны,
Гряди на подвиг славный, и спасай
Для жизни вечной человечий род".

Так пели победившему хвалу
Спасителю; скончав же чудный пир,
Сказали путь домой, и вскоре Он
Под Материнский воротился кров.
```

Подобно head, команда `tail` включает между файлами разделители, упрощающие работу с информацией.

Просмотр произвольного числа последних строк из файлов

```
tail -n
```

Продолжая аналогии между head и `tail`, можно предположить наличие опции, посредством которой задается число строк, предназначенных для отображения. Действительно,

такая опция существует. Это опция -n (или --lines=). По умолчанию отображается десять строк. Хотите увидеть окончания сразу нескольких файлов? Укажите их при вызове команды.

```
$ tail -n 4 Paradise_Lost.txt
  Paradise_Regained.txt
==> Paradise_Lost.txt <==
Ведомые, шагая тяжело,
Как странники, они рука в руке,
Эдем пересекая, побрели
Пустынною дорогою своей.
==> Paradise_Regained.txt <==
Так пели победившему хвалу
Спасителю; скончав же чудный пир,
Сказали путь домой, и вскоре Он
Под Материнский воротился кров.
```

Данный вариант команды `tail` удобен в тех случаях, когда надо просмотреть несколько файлов протоколов. Однако есть и лучшее решение. Вы узнаете о нем из следующего раздела.

Просмотр обновляемых строк в конце файла

```
tail -f
tail -f --pid=идентификатор
```

Файлы протоколов претерпевают постоянные изменения. Команда `tail` дает "мгновенный снимок" файла, а затем возвращает управление командной строке. А если вам надо еще раз обратиться к файлу протокола? Необходимо снова вызвать команду `tail`, затем еще и еще раз.

Если при вызове команды `tail` указать опцию `-f` (или `--follow`), программа не завершит работу. Она будет отображать последние десять строк (или другое их количество,

заданное посредством опции -n), причем изменения будут сразу же представлены на экране. Такой режим чрезвычайно удобен, если вам надо выявить причину некорректной работы программы или всей системы.

Например, файл протокола Web-сервера может выглядеть следующим образом:

```
$ tail -f /var/log/httpd/d20srd_org_log_20051201
"GET /srd/skills/bluff.htm HTTP/1.1"...
"GET /srd/skills/senseMotive.htm HTTP/1.1"...
"GET /srd/skills/concentration.htm HTTP/1.1"...
"GET /srd/classes/monk.htm HTTP/1.1"...
"GET /srd/skills/escapeArtist.htm HTTP/1.1"...
```

На заметку

Для экономии места я удалил IP-адрес, дату и время.

В книге трудно представить тот факт, что файл не закрывается. Программа `tail` продолжает работу с файлом и отражает изменения его содержимого. Очередные данные будут выводиться до тех пор, пока вы не нажмете комбинацию клавиш <Ctrl+C>, завершив тем самым работу программы.

Попробуйте применить данную команду к одному из ваших файлов протоколов, например `/var/log/syslog`. Задайте отображение требуемого количества строк, а затем попробуйте поработать с двумя файлами, например `/var/log/syslog` и `/var/log/daemon.log`, и посмотрите, что произойдет. Результаты могут оказаться совсем неожиданными.

Выводы

В данной главе мы рассмотрели четыре команды: `cat`, `less`, `head` и `tail`. Они обеспечивают просмотр текстовых файлов различными способами. Команда `cat` выводит сразу весь файл, команда `less` отображает информацию по страницам. Команды `head` и `tail` представляют "две стороны одной медали" или, вернее, одного файла. Команда `head` отображает его начало, а команда `tail` — последние строки. Совместно эти четыре команды дают возможность просмотреть любые части текстового файла.

6

Вывод на печать

В разное время в системе Linux использовались различные системы печати, в том числе всем известные LPD (Line Printer Daemon) и LPRng (LPR Next Generation), которые до сих пор встречаются в некоторых современных версиях Linux. В последние несколько лет в большинстве версий Linux применяется CUPS — Common Unix Printing System. CUPS — современная, простая в использовании высококачественная система, способная заменить LPD и LPRng. Команды, типичные для LPD и LPRng, по-прежнему доступны, но для их выполнения вызываются функции CUPS.

В этой главе мы рассмотрим систему CUPS, так как именно с ней в настоящее время работает большая часть пользователей Linux. Мы не будем здесь обсуждать вопросы настройки принтера. В состав большинства дистрибутивных пакетов входят инструменты конфигурирования с графическим пользовательским интерфейсом, поэтому мы сосредоточим внимание на обращении к устройству печати из командной строки.

На заметку

В статье *Overview of Linux Printing Systems*, опубликованной в журнале Linux Journal, приведен обзор некоторых возможностей системы Linux, причем основное внимание уделено системе CUPS. Данная статья доступна по адресу `http://www.linuxjournal.com/article/6729`. Подробно о CUPS рассказано в статье *The CUPS Printing System* того же журнала, которую можно скопировать, обратившись по адресу `http://www.linuxjournal.com/article/8618`. Конечно же, самые полные сведения о CUPS содержатся в руководстве *CUPS Software Users Manual* (`http://www.cups.org/doc-1.1/sum.html`). Объем описания велик, в некоторых случаях оно сложно для восприятия, но в нем содержатся полезные советы и ссылки на многочисленные ресурсы.

Получение списка доступных принтеров

```
lpstat -p
```

Приступая к работе с принтерами, надо знать, какие из них подключены к системе. Эту информацию предоставит команда `lpstat` (сокращение от line printer status), при вызове которой надо задать опцию `-p`.

```
$ lpstat -p
printer bro is idle. enabled since Jan 01 00:00
printer bro_wk is idle. enabled since Jan 01 00:00
printer wu is idle. enabled since Jan 01 00:00
```

Как видите, в данном случае к системе подключены три принтера, `bro`, `bro_wk` и `wu`, причем ни один из них в данный момент не занят. Вам может показаться странным, что все три принтера были доступны с полуночи 1 января, но в данный момент эту особенность можно проигнорировать.

Определение принтера по умолчанию

```
lpstat -d
```

Благодаря команде `lpstat -p` мы знаем все наши принтеры, но какой из них используется по умолчанию? Как вы вскоре увидите, задачу печати можно послать на конкретный принтер, однако проще всего использовать принтер по умолчанию. Для того чтобы выяснить, какой из принтеров задан по умолчанию, надо указать при вызове команды `lpstat` опцию `-d` (сокращение от `default`).

```
$ lpstat -d
system default destination: bro
```

Если к системе подключен только один принтер, то данная команда излишня. Однако если вы работаете с портативным компьютером и вам приходится выводить информацию на различные принтеры, команда `lpstat -d` придется как нельзя кстати.

Определение расположения принтеров

```
lpstat -s
```

Данная команда очень полезна для пользователей портативных компьютеров, так как она сообщает, как обратиться к доступным принтерам. Когда вы впервые подключите принтер, вам надо указать, как взаимодействовать с ним. Возможно несколько вариантов.

- Локальное подключение (через параллельный, последовательный или USB-порт).
- Удаленная очередь LPD.
- Разделяемый принтер SMB (Windows).
- Сетевой принтер (TCP).

- Удаленный сервер CUPS (IPP/HTTP).
- Сетевой принтер, использующий IPP (IPP/HTTP).

Для того чтобы выяснить как настроены принтеры для взаимодействия с вашим компьютером и как работать с ними, надо выполнить команду `lpstat` с опцией `-s`.

```
$ lpstat -s
system default destination: bro
device for bro: socket://192.168.0.160:9100
device for bro_wk: socket://192.168.1.10:9100
device for wu: socket://128.252.93.10:9100
```

В данном случае каждый принтер является сетевым и идентифицируется как `socket://`, за которым следует IP-адрес и порт (для большинства сетевых принтеров стандартным является порт 9100, однако в ряде случаев используется также порт 35). Пока все достаточно просто, однако ситуация может усложниться.

Несмотря на то что система CUPS дружественна по отношению к пользователям, если для идентификации принтера применяется URI (Uniform Resource Indicator), могут возникать проблемы. В табл. 6.1 приводятся методы соединения и типы URI. Сведения, содержащиеся в таблице, позволят вам лучше разобраться с данными, полученными в результате выполнения команды `lpstat -s`.

На заметку

Здесь предполагается, что принтер имеет имя `bro` и подключен по адресу 192.168.0.160. Эта информация применима не во всех случаях: если принтер присоединен через параллельное подключение, IP-адрес не имеет смысла.

Таблица 6.1. Методы соединения принтера и URI CUPS

Метод соединения	Пример URI (принтер bro имеет адрес 192.168.0.160)
Параллельное	`parallel:/dev/lp0`
Последовательное	`serial:/dev/ttyS1?baud=115200`
USB	`usb:/dev/usb/lp0`
Удаленная очередь LPD	`lpd://192.168.0.160/LPT1`
Разделяемый принтер SMB (Windows)	`smb://`*`пользовательское_имя`*`:`*`пароль`*`@192.168.0.160/bro`
Сетевой принтер (TCP)	`socket://192.168.0.160:9100`
Удаленный сервер CUPS (IPP/HTTP)	`ipp://192.168.0.160:631/printers/bro,http://192.168.0.160/printers/bro`
Сетевой принтер с IPP (IPP/HTTP)	`ipp://192.168.0.160:631/printers/bro,http://192.168.0.160/printers/bro`

Благодаря развитию структуры сетевых принтеров упростилось взаимодействие с ними. В частности, обмен данными возможен через сокет, средствами IPP или HTTP. Однако не исключено, что вам попадется принтер, использующийся достаточно давно. Вполне возможно, что для взаимодействия с ним применяются устаревшие методы, поэтому желательно знать их.

Совет

Команда `lpstat -s` дублирует команду `lpstat -p -d`, т.е. предоставляет список всех принтеров, известных системе, и сообщает о принтере по умолчанию. Данная команда удобна для быстрого получения нужной информации.

Получение полной информации о принтерах

```
lpstat -t
```

Команды `lpstat -p`, `lpstat -d` и `lpstat -s` хороши тем, что они предоставляют конкретные сведения. Если же вы хотите получить сразу всю информацию о принтерах, вам надо задать при вызове `lpstat` опцию `-t`.

```
$ lpstat -t
scheduler is running
system default destination: bro
device for bro: socket://192.168.0.160:9100
device for bro_wk: socket://192.168.1.10:9100
device for wu: socket://128.252.93.10:9100
bro accepting requests since Jan 01 00:00
bro_wk accepting requests since Jan 01 00:00
wu accepting requests since Jan 01 00:00
printer bro is idle. enabled since Jan 01 00:00
printer bro_wk is idle. enabled since Jan 01 00:00
printer wu is idle. enabled since Jan 01 00:00
```

Вы получаете подробную информацию: принтер по умолчанию, список всех принтеров, известных системе, расположение принтеров и способы взаимодействия с ними, а также состояние принтеров. Чем больше устройств печати имеется в системе, тем длиннее будет список. Если вам покажется, что объем информации слишком велик, вы можете вызвать команду `lpstat` с одной из опций, которые обсуждались ранее.

Вывод информации на принтер по умолчанию

```
lpr
```

Теперь, когда вы имеете сведения о принтерах, имеющихся в системе, надо использовать их для вывода информации на печать. Вывести данные на принтер по умолчанию (определяемый посредством `lpstat -d`) несложно.

```
$ lpr Lovecraft_-_Call_of_Cthulhu.txt
```

Достаточно ввести команду `lpr` и имя текстового файла.

На заметку

Вероятно, вы считаете, что данные, пригодные для вывода на принтер, ограничиваются текстовыми ASCII-файлами. Если это так, то вы будете приятно удивлены, узнав, что можно также выводить на печать файлы PDF и PostScript. Но это все! Не пытайтесь распечатать документы Word, OpenOffice.org, в общем, любые файлы, отличные от текстовых и PostScript-документов, иначе вы получите множество страниц, заполненных непонятными символами.

Вывод информации на произвольно выбранный принтер

```
lpr -P
```

Как было показано в предыдущем разделе, вывод данных на принтер по умолчанию — простая задача. Если в вашем распоряжении есть несколько принтеров и если вы хотите вывести информацию на устройство печати, отличное от устройства, заданного по умолчанию, вам надо задать опцию `-P`, а затем имя принтера.

```
$ lpr -P bro_wk Lovecraft_-_Call_of_Cthulhu.txt
```

Если вы не знаете имен принтеров, выполните команду `lpstat -p`, которая обсуждалась ранее в этой главе.

На заметку

Вы, вероятно, заметили, что вместо пробелов в имени файла использованы знаки подчеркивания. Это упрощает параметра команды. Если бы в имени были пробелы, его пришлось бы указывать одним из следующих способов:

```
$ lpr -P bro_wk "Lovecraft - Call of Cthulhu.txt"
```

или

```
$ lpr -P bro_wk Lovecraft\ -\ Call\ of\ Cthulhu.txt
```

При наличии пробелов в имени проще всего использовать заполнение по табуляции, а не вводить имя вручную. Например, если вы введете `lpr -P bro_websanity Love`, а затем нажмете клавишу <Tab>, оболочка самостоятельно завершит ввод имени файла. Подробнее о завершении ввода по табуляции можно прочитать в документе, который находится по адресу `http://www.slackbook.org/html/shell-bash.html#SHELL-BASH-TAB`.

Вывод нескольких копий файла

```
lpr -#
```

Если вам надо вывести несколько копий одного документа, используйте опцию `-#`, указав после нее число копий.

```
$ lpr -# 2 -P bro Lovecraft_-_Call_of_Cthulhu.txt
```

Число копий можно задавать в диапазоне от 1 до 100. Если вам требуется более 100 копий, повторите команду или напишите сценарий. А еще лучше в этом случае выполнить работу на специализированном полиграфическом оборудовании.

Получение списка заданий на печать

```
lpq
```

Если вы послали на печать несколько заданий, вам может потребоваться информация о них. Возможно, вы захотите отменить одно из заданий (подробнее этот вопрос будет рассмотрен в последующих разделах), или окажется, что выполнение какого-то задания занимает слишком много времени. А может быть, вы просто захотите узнать, как обстоит дело с печатью. Команда `lpq` (сокращение от lp queue) выводит информацию о всех заданиях, предназначенных для вывода на принтер по умолчанию (как вы уже знаете, сведения об этом принтере можно получить по команде `lpstat -d`).

```
$ lpq
bro is ready and printing
Rank    Owner   Job  File(s)
   Total Size
active  scott   489  Lovecraft_-_Call_of_C
  108544 bytes
```

Если вам надо выяснить состояние очередей для всех принтеров, а не только для устройства, заданного по умолчанию, добавьте после `lpq` опцию `-a` (сокращение от all).

```
$ lpq -a
Rank    Owner   Job  File(s)
   Total Size
active  scott   489  Lovecraft_-_Call_of_C
  108544 bytes
1st     scott   490  ERB_-_A Princess_of_M
  524288 bytes
```

Заметьте следующее. Во-первых, информация, предоставляемая посредством `lpq -a`, иногда усекается, поэтому,

если файл имеет имя Lovecraft_-_Call_of_Cthulhu.txt или ERB_-_A Princess_of_Mars.txt, мы увидим не все имя, а лишь ограниченное число символов. Во-вторых, lpq сообщает не о всех заданиях, известных принтеру, а лишь о тех, о которых знает ваш компьютер. С точки зрения принтера очередь может выглядеть приблизительно так:

- Lovecraft_-_Call_of_Cthulhu.txt
- Doyle_-_The_Lost_World.txt
- ERB_-_A Princess_of_Mars.txt

Для того чтобы выяснить реальное состояние очереди, вам надо использовать одну из утилит, ориентированных на конкретное устройство, однако рассмотрение этого вопроса выходит за рамки данной книги.

Вывод информации о заданиях для конкретного принтера

```
lpstat
```

Команда lpq предоставляет информацию о файлах, поставленных в очередь на печать, но ничего не сообщает о том, на какой принтер они должны быть выведены. Для того чтобы выяснить это, надо использовать команду lpstat, которая рассматривалась ранее в этой главе. На этот раз мы введем lpstat без опций.

```
$lpstat
bro-489      rsgranne  108544  Tue 10 Dec 2005
bro_wk-490   rsgranne  524288  Tue 10 Dec 2005
```

В результате будет получен список заданий; в начале каждой строки указано имя принтера, которому соответствует конкретное задание. Может случиться так, что вы передали задание на принтер, который в текущий момент

не подключен. В этом случае вам надо выполнить команду `lpstat`, а затем удалить задание. Как это сделать, вы узнаете в следующем разделе.

Отмена задания, переданного на принтер по умолчанию

```
lprm
```

Если вы хотите отменить задание, переданное на принтер по умолчанию, используйте команду `lprm` (сокращение от lp remove).

```
$ lprm
```

Не медлите перед вводом этой команды. Быстродействие современных принтеров настолько велико, и они содержат настолько большой объем памяти, что задание может быстро покинуть ваш компьютер и оказаться в памяти принтера. Если это случилось, найдите на принтере кнопку отмены и поскорее нажмите ее.

Отмена задания, переданного на произвольный принтер

```
lprm идентификатор_задания
```

В предыдущем разделе мы выяснили, как отменить текущее задание, переданное на принтер по умолчанию. Но что если задание уже было помещено в очередь и выполнение его не начнется в течение нескольких минут? Или если задание передано на устройство печати, не назначенное по умолчанию? В этих случаях вы также можете использовать `lprm`, но необходимо сообщить команде, какое задание должно быть отменено, задав идентификатор задания.

Вернемся к примеру, рассмотренному при обсуждении вопроса о получении информации о заданиях на печать. В третьем столбце под названием Job содержится число. В разделе, в котором рассматривался вопрос получения информации о заданиях каждому принтеру, также был листинг; в нем за именем принтера следовал дефис, а за ним число. И в том и в другом случае это идентификационный номер задания на печать. Введите этот номер при вызове lprm, и соответствующее задание будет удалено.

```
$lpstat
bro-489      rsgranne  108544  Tue 10 Dec 2005
bro_wk-490   rsgranne  524288  Tue 10 Dec 2005
$ lprm 490
$lpstat
bro-489      rsgranne  108544  Tue 10 Dec 2005
```

Теперь, в случае ошибки, вам не придется зря расходовать бумагу и тонер. Выяснив, что вы случайно передали на печать документ объемом в 500 страниц с изображениями, вы отмените задание, вызвав команду lprm и указав идентификатор.

Отмена всех заданий на печать

```
lprm -
```

Что делать, если вам надо отменить несколько заданий? Вы можете сделать это, указав в командной строке идентификационные номера, например, так, как показано ниже.

```
$ lprm 489 490 491 492 493
```

Однако при этом надо вводить много чисел. Если вы хотите избавиться сразу от всех заданий, независимо от того, на какой принтер вы отправили каждый из них, надо ввести после lprm символ -.

```
$ lprm -
```

Это быстро, не требует усилий, одним словом, работа для лентяев в традициях Linux. И в этом нет ничего зазорного!

Выводы

Вывод на печать — одна из задач, которые постоянно приходится решать при работе на компьютере, поэтому важно знать, как делать это наиболее эффективно. Прежде всего, надо знать, как обращаться к принтерам, и уметь передавать задания на печать. Задавая печать файла, можно ошибиться, поэтому необходимо знать, как отменить то или иное задание. В этой главе рассмотрены основные вопросы взаимодействия с принтерами из системы Linux. И в завершение хочу сказать, что в последнее время существенно подешевели устройства двусторонней печати, поэтому если вы приобретете такое устройство, то сократите расход бумаги.

7

Владельцы файлов и права доступа

Система Unix, а следовательно, и Linux разрабатывалась как многопользовательская система (в отличие от нее, Windows была изначально ориентирована на одного пользователя, отсюда и проблемы с защитой). Это означает, что с системой могут одновременно работать несколько пользователей: создавать файлы, удалять каталоги, просматривать текстовые файлы и выполнять другие, самые разнообразные действия. Для того чтобы пользователи не мешали друг другу и тем более не повредили операционную систему, был разработан механизм прав доступа. Понимание прав доступа существенно упрощает использование Linux, независимо от того, установлена ли эта система на рабочей станции, ориентированной на одного пользователя, или на сервере, посещаемом многими. Несмотря на то что данный инструмент довольно простой, он очень мощный. Попробуем разобраться с ним.

Изменение групп для файлов и каталогов

`chgrp`

Когда вы, работая в системе Linux, создаете новый файл (или каталог), владельцами его назначаетесь вы и группа, в которую вы входите. Такое поведение реализуется по умолчанию практически во всех системах. Предположим, например, что вы собираетесь написать новый сценарий.

На заметку

Для экономии места информация, которая обычно отображается по команде `ls -l`, заменена многоточием.

```
$ touch new_script.sh
$ ls -l
-rw-r--r-- 1 scott scott ... script.sh
```

На заметку

В данном случае и пользователь, и группа имеет имя `scott`, однако это не обязательно должно быть так. При создании файла идентификатор пользователя используется для обозначения владельца файла, а идентификатор группы — для обозначения группы.

Предположим также, что вы входите в группу, называемую `admins`, и хотите, чтобы этот сценарий могли запускать другие члены данной группы. Как добиться этого? Заменить группу `scott` на `admins` можно с помощью команды `chgrp`.

```
$ chgrp admins new_script.sh
$ ls -l
-rw-r--r-- 1 scott admins ... script.sh
```

На заметку

Да, действительно, сценарий не будет работать, так как файл, в котором он содержится, не является исполняемым. Как исправить ситуацию, вы узнаете далее в этой главе при рассмотрении команды chmod.

Говоря о команде chgrp, необходимо отметить две особенности. Во-первых, при вызове команды chgrp можно указывать либо имя группы, либо ее идентификатор. Как же выяснить, какой идентификатор соответствует той или иной группе? Проще всего сделать это, применив команду cat к файлу /etc/group, содержащему информацию о группах.

```
$ cat /etc/group
bind:x:118:
scott:x:1001:
admins:x:1002:scott,alice,bob
[Данные сокращены для экономии места]
```

Во-вторых, пытаясь применить данную команду, следует учитывать вопросы безопасности. Изменять принадлежность группе можно только в том случае, если вы являетесь ее членом. Другими словами, пользователи scott, alice и bob могут использовать команду chgrp, чтобы установить принадлежность файла или каталога группе admins, а carol не может сделать этого, поскольку она не является членом группы admins.

Рекурсивное изменение принадлежности каталога группе

```
chgrp -R
```

В ряде случаев нецелесообразно изменять принадлежность группе лишь для одного файла или каталога. Символы групповых операций позволяют сделать это сразу

для нескольких файлов, содержащихся в каталоге. Если вы хотите изменить группу для каталога и всего начинающегося с него поддерева файловой системы, вам надо использовать опцию -R (или --recursive).

```
$ pwd
/home/scott/pictures/libby
$ ls -F
by_pool/ libby_arrowrock.jpg libby.jpg on_floor/
$ ls -lF *
-rw-r--r-- 1 scott scott ... libby_arrowrock.jpg
-rw-r--r-- 1 scott scott ... libby.jpg

by_pool/:
-rw-r--r-- 1 scott scott ... libby_by_pool_02.jpg
drwxr-xr-x 2 scott scott ... lieberman_pool

on_floor/:
-rw-r--r-- 1 scott scott ...
  libby_on_floor_01.jpg
-rw-r--r-- 1 scott scott ...
  libby_on_floor_02.jpg
$ chgrp -R family */*
$ ls -l *
-rw-r--r-- 1 scott family ... libby_arrowrock.jpg
-rw-r--r-- 1 scott family ... libby.jpg

by_pool:
-rw-r--r-- 1 scott family ...
  libby_by_pool_02.jpg
drwxr-xr-x 2 scott family ... lieberman_pool

on_floor:
-rw-r--r-- 1 scott family ...
  libby_on_floor_01.jpg
-rw-r--r-- 1 scott family ...
  libby_on_floor_02.jpg
```

Внимание!

Если вы выполните команду `chgrp -R family *`, она не затронет файлов, имена которых начинаются с точки (в данном примере речь идет о каталоге `/home/scott/pictures/libby`). Однако все произойдет по-другому, если команда будет выглядеть так: `chgrp -R family .*`. Она не только изменит принадлежность группе файлов текущего каталога, начинающихся с точки. Имя `..` также соответствует шаблону `.*`, поэтому изменения коснутся также файлов, находящихся в родительском каталоге. Вряд ли вы стремились получить такой результат!

Отслеживание изменений, которые вносятся посредством команды chgrp

```
chgrp -v
chgrp -c
```

Вероятно, вы заметили, что `chgrp`, как и большинство других программ, работающих под управлением Linux, отображает какую-либо информацию только в случае возникновения проблем. Если программа работает корректно, вы не получите сообщения типа "все идет по плану". Информация в командной строке появляется только в том случае, когда требуется вмешательство пользователя в ход процесса.

Ели вы хотите получать подробные сведения о действиях, выполняемых `chgrp`, вам надо задать опцию `-v` (или `--verbose`). При этом на каждом этапе своей работы программа `chgrp` будет выдавать вам подробные сообщения.

```
$ ls -lF
drwxr-xr-x 4 scott scott ... by_pool/
-rw-r--r-- 1 scott scott ... libby_arrowrock.jpg
-rw-r--r-- 1 scott family ... libby.jpg
```

```
-rw-r--r-- 1 scott scott ... libby_on_couch.jpg
drwxr-xr-x 2 scott scott ... on_floor/
$ chgrp -v family *
changed group of 'by_pool' to family
changed group of 'libby_arrowrock.jpg' to family
group of 'libby.jpg' retained as family
changed group of 'libby_on_couch.jpg' to family
changed group of 'on_floor' to family
$ ls -lF
drwxr-xr-x 4 scott family ... by_pool/
-rw-r--r-- 1 scott family ... libby_arrowrock.jpg
-rw-r--r-- 1 scott family ... libby.jpg
-rw-r--r-- 1 scott family ... libby_on_couch.jpg
drwxr-xr-x 2 scott family ... on_floor/
```

Обратите внимание на особенности данного примера. Файл `libby.jpg` и ранее принадлежал группе `family`, однако, поскольку вы задали опцию `-v`, программа проинформировала вас о том, что файл `libby.jpg` по-прежнему принадлежит группе `family` (`group of 'libby.jpg' retained as family`). Изменяя группу, вы вряд ли нуждаетесь в информации о файлах, которые и ранее принадлежали вновь заданной группе. Избавиться от ненужной информации позволит опция `-c` (или `--changes`), которая задает вывод информации только о реальных изменениях.

```
$ ls -lF
drwxr-xr-x 4 scott scott ... by_pool/
-rw-r--r-- 1 scott scott ... libby_arrowrock.jpg
-rw-r--r-- 1 scott family ... libby.jpg
-rw-r--r-- 1 scott scott ... libby_on_couch.jpg
drwxr-xr-x 2 scott scott ... on_floor/
$ chgrp -c family *
changed group of 'by_pool' to family
changed group of 'libby_arrowrock.jpg' to family
```

```
changed group of 'libby_on_couch.jpg' to family
changed group of 'on_floor' to family
$ ls -lF
drwxr-xr-x 4 scott family ... by_pool/
-rw-r--r-- 1 scott family ... libby_arrowrock.jpg
-rw-r--r-- 1 scott family ... libby.jpg
-rw-r--r-- 1 scott family ... libby_on_couch.jpg
drwxr-xr-x 2 scott family ... on_floor/
```

На этот раз сообщение о файле `libby.jpg` отсутствует, так как он и ранее принадлежал группе `family`. Таким образом, если вам нужен полный отчет, включающий даже информацию о файлах, не подвергшихся изменениям, используйте опцию `-v`; если же вас интересуют только реальные изменения, вам надо задать опцию `-c`.

Изменение владельцев файлов и каталогов

`chown`

Возможность изменить группу для файла достаточно важна, но гораздо чаще вы будете сталкиваться с задачей изменения владельца этого файла. Если для изменения группы применяется команда `chgrp`, то для изменения владельца надо воспользоваться командой `chown`.

```
$ ls -l
-rw-r--r-- 1 scott scott ... libby_arrowrock.jpg
-rw-r--r-- 1 scott family ... libby.jpg
-rw-r--r-- 1 scott scott ... libby_on_couch.jpg
$ chown denise libby.jpg
$ ls -l
-rw-r--r-- 1 scott scott ... libby_arrowrock.jpg
-rw-r--r-- 1 denise family ... libby.jpg
-rw-r--r-- 1 scott scott ... libby_on_couch.jpg
```

Некоторые сведения о chgrp, изложенные ранее, справедливы также и для команды chown. Команда chown использует либо имя пользователя, либо его числовой идентификатор. Выяснить числовой идентификатор можно с помощью команды cat /etc/passwd, которая предоставит информацию наподобие следующей:

```
bind:x:110:118::/var/cache/bind:/bin/false
scott:x:1001:1001:Scott,,,:/home/scott:/bin/bash
ntop:x:120:120::/var/lib/ntop:/bin/false
```

Первое из чисел, встречающихся в строке, — это числовой идентификатор пользователя (второе число — идентификатор основной группы, в состав которой входит пользователь).

Изменить владельца файла можно только в том случае, если вы зарегистрировались под именем владельца либо являетесь пользователем root. Хотя такое ограничение очевидно, некоторые пользователи забывают о нем.

Внимание!

Если вы используете команду chown -R scott *, вы не измените файлы, содержащиеся в текущем каталоге, имена которых начинаются с точки. Однако все произойдет по-другому, если команда будет выглядеть так: chown -R scott .*. Она не только изменит принадлежность файлов текущего каталога, начинающихся с точки. Имя .. также соответствует шаблону .*, поэтому изменения коснутся и файлов, находящихся в родительском каталоге. Вряд ли вы стремились получить такой результат!

Изменение владельца и группы для файлов и каталогов

```
chown владелец:группа
```

Вы уже знаете, что команда chgrp позволяет изменить группу, а команда chown — владельца. С помощью команды chown можно также решить обе эти задачи одновременно. Для этого надо в качестве параметра chown задать имя пользователя, а за ним имя группы, разделив их двоеточием (такой формат команды chown — одна из причин, по которым не рекомендуется включать двоеточие в имя пользователя или группы). После этого, как обычно, следует имя файла или каталога.

```
$ ls -l
-rw-r--r-- 1 scott scott ... libby.jpg
$ chown denise:family libby.jpg
$ ls -l
-rw-r--r-- 1 denise family ... libby.jpg
```

Вы можете даже изменить с помощью команды chown только группу. Для этого надо не указывать перед двоеточием имя пользователя.

```
$ ls -l
-rw-r--r-- 1 scott scott ... libby.jpg
$ chown :family libby.jpg
$ ls -l
-rw-r--r-- 1 scott family ... libby.jpg
```

Совет

Что делать, если в имени пользователя или группы есть двоеточие? В этом случае надо предварять данный символ обратной косой чертой, что отменяет его специальное значение. Двоеточие, перед которым указана обратная косая черта интерпретируется как обычный

знак, а не как разделитель между именем пользователя и именем группы.

```
$ chown denise:family\:parents libby.jpg
```

Данный подход позволяет справиться с проблемой, но гораздо лучше будет, если вы вовсе откажетесь от использования двоеточия в имени пользователя или группы.

Поскольку команда chown позволяет выполнять те же функции, что и chgrp, последнюю можно не использовать вовсе.

На заметку

Для разделения пользователя и группы можно применять либо точку, либо двоеточие, однако рекомендуется все же отдать предпочтение двоеточию. Точка в качестве разделителя считается устаревшим форматом и не рекомендована к применению.

Общие сведения о правах доступа

Перед тем как приступить к изучению команды chmod, позволяющей изменять права доступа, соответствующие файлу или каталогу, рассмотрим, как система Linux интерпретирует эти права.

На заметку

В настоящее время готовится переход системы Linux на использование более гибкой и мощной системы прав доступа, называемой ACL (Access Control Lists). Сейчас ACL используется достаточно редко, поэтому мы не будем рассматривать здесь эту систему. Дополнительную информацию об ACL можно найти в статье *Access Control Lists*, опубликованной в журнале *Linux Magazine* (она доступна по адресу `http://www.linux-mag.com/2004-11/guru_01.html`) или в документе *An ACL GUI for Linux* (`http://opensource.weblogsinc.com/2005/12/06/an-acl-gui-for-linux/`).

Для каждого файла или каталога Linux различает три категории пользователей: владелец, группа и все остальные пользователи системы. Эти категории перечислены в табл. 7.1; там же указаны буквы, применяющиеся для их обозначения.

Таблица 7.1. Категории пользователей и их обозначение

Категория пользователей	Сокращенное обозначение
Владелец	u
Группа	g
Прочие пользователи	o

В главе 2 мы говорили о правах, указывающих, какие пользователи могут работать с файлами и каталогами. В этом разделе речь пойдет о трех атрибутах, соответствующих чтению, записи и выполнению. Для их обозначения применяются буквы `r`, `w` и `x`. Кроме того, существуют также признаки `suid`, `sgid` и специальный бит, называемый "sticky bit". Они представляются соответственно буквами `s` (или в некоторых случаях `S`), `s` (или `S`) и `t` (или `T`). Учтите, что эти признаки имеют различное значение в зависимости от того, относятся ли они к файлу или каталогу. В табл. 7.2 перечислены атрибуты, их сокращенное представление и значение.

Таблица 7.2. Атрибуты, определяющие доступ

Атрибут	Сокращенное представление	Значение для файла	Значение для каталога
Чтение	`r`	Можно читать	Можно просматривать содержимое с помощью команды `ls`

Окончание табл. 7.2

Атрибут	Сокращенное представление	Значение для файла	Значение для каталога
Запись	w	Можно редактировать	Можно удалять, переименовывать или добавлять файлы
Выполнение	x	Можно запускать на выполнение	Можно читать файлы и каталоги и запускать файлы на выполнение
suid	s	Любой пользователь может запустить файл на выполнение с правами его владельца	Не применяется
sgid	s	Любой пользователь может запустить файл на выполнение с правами группы	Все файлы, вновь создаваемые в каталоге, принадлежат группе, владеющей каталогом
"sticky bit"	t	Сообщает системе о том, что файл часто используется и должен находиться в области подкачки для быстрого обращения (применяется в старых версиях Unix, в системе Linux игнорируется)	Удалять или переименовывать файлы, находящиеся в каталоге, имеют право только их владельцы или владелец каталога

На заметку

Пользователь root может выполнять любые действия с любым файлом или каталогом, поэтому сведения из табл. 7.2 неприменимы к нему.

В последующих разделах мы подробнее рассмотрим указанные атрибуты. Теперь, когда вы знакомы с основами, обсудим использование команды chmod для изменения прав доступа к файлам и каталогам.

Изменения прав доступа к файлам и каталогам с использованием символьных обозначений

```
chmod [ugo][+-=][rwx]
```

Для команды chmod предусмотрены два типа обозначений: символьные и числовые. Оба имеют свои преимущества, однако для пользователя проще сначала изучить символьное представление. Для формирования символьных обозначений используется простой принцип: категория пользователей, на которую необходимо воздействовать, затем символ "плюс" (+) для назначения, "минус" (-) для удаления или знак равенства (=) для конкретной установки прав, а затем буквы (r, w, x, s, t), представляющие права, которые необходимо изменить. Предположим, например, что вам надо предоставить членам группы family право модифицировать изображение.

```
$ ls -l
-rw-r--r-- 1 scott family ... libby.jpg
$ chmod g+w libby.jpg
$ ls -l
-rw-rw-r-- 1 scott family ... libby.jpg
```

Как видите, это достаточно просто. А что делать, если вы хотите дать право на запись в файл не только членам группы family, но и всем остальным пользователям?

```
$ ls -l
-rw-r--r-- 1 scott family ... libby.jpg
$ chmod go+w libby.jpg
$ ls -l
-rw-rw-rw- 1 scott family ... libby.jpg
```

Предоставляя всем пользователям — владельцу, группе и остальным — права чтения и записи, вы можете также сделать это и по-другому.

```
$ ls -l
-rw-r--r-- 1 scott family ... libby.jpg
$ chmod a=rw libby.jpg
$ ls -l
-rw-rw-rw- 1 scott family ... libby.jpg
```

Предположим, что вы заметили ошибку и хотите лишить группу family и всех остальных пользователей права изменять изображение, более того, вы хотите сделать так, чтобы пользователи, не являющиеся владельцем и членами группы, не могли даже просматривать картинку.

```
$ ls -l
-rw-rw-rw- 1 scott family ... libby.jpg
$ chmod go-w libby.jpg
$ ls -l
-rw-r--r-- 1 scott family ... libby.jpg
$ chmod o-r libby.jpg
$ ls -l
-rw-r----- 1 scott family ... libby.jpg
```

Вместо символа - можно использовать знак равенства.

```
$ ls -l
-rw-rw-rw- 1 scott family ... libby.jpg
$ chmod g=r libby.jpg
$ ls -l
-rw-r--rw- 1 scott family ... libby.jpg
$ chmod o= libby.jpg
$ ls -l
-rw-r----- 1 scott family ... libby.jpg
```

Заметьте, что в последней команде chmod после знака равенства, следующего за o, не указан никакой символ. Этим удаляются все права для остальных пользователей системы. Как видите, необходимый результат достигается быстро и эффективно.

Преимущество символьных обозначений состоит в том, что пользователь может быстро освоить их. Основной недостаток виден из последнего примера: если вы хотите внести разные изменения для двух групп пользователей, вам надо запустить chmod два раза. Данную проблему можно устранить посредством числовых обозначений, которые будут рассмотрены в следующем разделе.

Изменения прав доступа к файлам и каталогам с использованием числовых обозначений

```
chmod [0-7][0-7][0-7]
```

Для числовых обозначений применяется восьмеричная система счисления. Мы не будем обсуждать, почему используются те или иные числа, к тому же изменить существующее положение дел не в наших силах. Рассмотрим лишь их значения. Праву на чтение (r) соответствует значение 4, праву на запись (w) — значение 2 и праву на выполнение (x) — значение 1. Как вы знаете, права доступа Linux

предполагают наличие трех категорий пользователей: владелец, группа и все остальные. Для каждой категории можно разрешить чтение, запись и выполнение так, как показано в табл. 7.3.

Таблица 7.3. Права доступа и их числовое представление

	Владелец	Группа	Остальные пользователи
Символьное представление	r; w; x	r; w; x	r; w; x
Числовое представление	4;2;1	4;2;1	4;2;1

Назначение прав по данной схеме сводится к обычному сложению чисел. Рассмотрим несколько примеров.

- Пользователь имеет право на чтение и запись в файл или каталог. Чтению соответствует значение 4, записи — 2. Поскольку права на выполнение нет, соответствующее значение равно нулю. 4 + 2 + 0 = 6.
- Пользователь имеет право читать файл и запускать его на выполнение. Чтению соответствует значение 4, записи, поскольку она запрещена, — 0. Выполнение обозначается числом 1. 4 + 0 + 1 = 5.
- Пользователь имеет право на чтение, запись и выполнение для каталога. Чтению соответствует значение 4, записи — 2, а выполнению — 1. 4 + 2 + 1 = 7.

Нетрудно заметить, что максимально возможное значение для одной категории пользователей равно 7 (чтение, запись и выполнение разрешены), а минимальное — 0 (чтение, запись и выполнение запрещены). Поскольку существуют три категории пользователей, права доступа задаются тремя цифрами; каждая из них принимает значение от 0 до 7. В табл. 7.4 представлены все допустимые числа для одной категории и их значения.

Таблица 7.4. Числовое представление прав доступа, отображаемых с помощью команды ls -l

Числовое представление	Представление ls -l
0	---
1	--x
2	-w-
3	-wx
4	r--
5	r-x
6	rw-
7	rwx

Несмотря на то что допустимо чрезвычайно большое число конкретных сочетаний прав, некоторые из них встречаются чаще других. Эти права перечислены в табл. 7.5; там же описано их значение.

Таблица 7.5. Часто встречающиеся сочетания прав

Команда chmod	Представление ls -l	Описание
chmod 400	-r--------	Пользователь имеет право чтения; никто другой не имеет права выполнять никакие действия
chmod 644	-rw-r--r--	Все пользователи имеют право чтения; владелец может редактировать
chmod 660	-rw-rw----	Владелец и группа могут читать и редактировать; остальные не имеют права выполнять никакие действия

Окончание табл. 7.5

Команда chmod	Представление ls -l	Описание
chmod 664	-rw-rw-r--	Все пользователи имеют право чтения: владелец и группа могут редактировать
chmod 700	-rwx------	Владелец может читать, записывать и запускать на выполнение; никто другой не имеет права выполнять никакие действия
chmod 744	-rwxr--r--	Каждый пользователь может читать; владелец имеет право редактировать и запускать на выполнение
chmod 755	-rwxr-xr-x	Каждый пользователь имеет право читать и запускать на выполнение; пользователь может редактировать
chmod 777	-rwxrwxrwx	Каждый пользователь может читать, редактировать и запускать на выполнение (устанавливать такой набор прав не рекомендуется)

Внимание!

В принципе для файла или каталога может быть выполнена команда chmod 000. В этом случае единственным пользователем, имеющим право на выполнение каких-либо действий, в том числе на вызов команды chmod, остается пользователь root.

Возможно, вам не совсем понятно, чем же хороши числовые обозначения. Действительно, чтобы разобраться в них,

надо приложить больше усилий, чем в случае с символьными обозначениями, однако они позволяют одновременно задавать любой набор прав. Вернемся к примерам назначения прав для файлов и каталогов, рассмотренных ранее, но вместо символьных используем числовые обозначения.

Предположим, например, что вам надо предоставить членам группы `family` право изменять изображение.

```
$ ls -l
-rw-r--r-- 1 scott family ... libby.jpg
$ chmod 664 libby.jpg
$ ls -l
-rw-rw-r-- 1 scott family ... libby.jpg
```

А что делать, если вы хотите предоставить право на запись в файл не только членам группы `family`, но и всем остальным пользователям?

```
$ ls -l
-rw-r--r-- 1 scott family ... libby.jpg
$ chmod 666 libby.jpg
$ ls -l
-rw-rw-rw- 1 scott family ... libby.jpg
```

Теперь вы заметили ошибку и хотите лишить группу `family` и всех остальных пользователей права изменять изображение, а также хотите сделать так, чтобы пользователи, не являющиеся владельцем и членами группы, не могли даже просматривать картинку.

```
$ ls -l
-rw-rw-rw- 1 scott family ... libby.jpg
$ chmod 640 libby.jpg
$ ls -l
-rw-r----- 1 scott family ... libby.jpg
```

На этих примерах хорошо видно основное преимущество числовых обозначений. Там, где при использовании символьных обозначений установка прав происходит в два этапа (`chmod go-w`, а затем `chmod o-r`, или `chmod g-r` и `chmod o=`), числовые обозначения позволяют обойтись одной командой. По этой причине специалисты, хорошо разбирающиеся в системе Linux, используют числовые обозначения, что позволяет им быстрее назначать требуемые права.

Рекурсивное изменение прав

```
chmod -R
```

Вы, вероятно, заметили, что многие команды Linux можно рекурсивно применять к файлам и каталогам. Команда `chmod` — не исключение. Опция `-R` (или `--recursive`) позволяет мгновенно воздействовать на сотни объектов файловой системы, надо лишь быть уверенным, что вам действительно необходимо изменить права для них всех.

```
$ pwd
/home/scott/pictures/libby

$ ls -lF
drwxrw---- 2 scott scott ... by_pool/
-rw-r--r-- 1 scott scott ... libby_arrowrock.jpg
-rw-r--r-- 1 scott scott ... libby.jpg
drwxrw---- 2 scott scott ... on_floor/
$ ls -l *
-rw-r--r-- 1 scott scott ... libby_arrowrock.jpg
-rw-r--r-- 1 scott scott ... libby.jpg

by_pool:
-rw-r--r-- 1 scott scott ... libby_by_pool_02.jpg
-rwxr-xr-x 2 scott scott ... lieberman_pool.jpg
```

```
on_floor:
-rw-r--r-- 1 scott scott ... libby_on_floor_01.jpg
-rw-r--r-- 1 scott scott ... libby_on_floor_02.jpg
$ chgrp -R family *
$ chmod -R 660 *
chmod: 'by_pool' : Permission denied
chmod: 'on_floor' : Permission denied
```

Однако в данном примере было получено сообщение `Permission denied`. Что случилось? Обратите внимание на табл. 7.2. Если файл является исполняемым, это означает, что его можно запустить как программу; для каталога же соответствующий признак указывает на то, что пользователи могут обращаться к его содержимому, читать файлы и подкаталоги.

Команда `chmod -R 660 *` удаляет право x и для файлов, и для каталогов, поэтому она не может сообщить о том, что задача выполнена: для каталогов уже сброшен признак x, и они недоступны.

Что же делать в этом случае? Найти ответ непросто. Можно использовать при вызове `chmod` символы групповых операций, но подбирать их так, чтобы команда воздействовала только на файлы определенного типа.

```
$ chmod -R 660 *.jpg
```

В данном примере права будут изменены только для изображений, но не для каталогов. Если в вашем распоряжении есть файлы различных типов, задача становится рутинной и отнимает много времени; вам придется выполнить команду `chmod` для каждого типа файлов.

Если в каталоге есть много подкаталогов или файлы различных типов и если вы приобрели достаточный опыт работы с Linux, вы можете использовать команду `find` для поиска файлов, не являющихся подкаталогами, а затем

изменять права только для них. Команда find будет подробно рассмотрена в главе 10.

Пока что можно ограничиться лишь следующей рекомендацией: изменяя права доступа рекурсивно, будьте осторожны. Вы можете получить неожиданный результат и больше не сможете обратиться к файлу или каталогу.

Установка и сброс suid

```
chmod u[+-]s
```

Ранее в этой главе мы рассматривали различные права доступа. Мы сосредоточили внимание на r, w и x, поскольку они применяются наиболее часто. Однако существуют и другие права, которые в ряде случаев могут быть очень полезны. Рассмотрим признак suid, который применяется для файлов, причем исполняемых. Для каталога признак suid не может быть указан.

Установленный признак suid означает, что пользователь может запустить файл на выполнение с правами пользователя, владеющего им, т.е. так, как будто этот файл был выполнен самим владельцем. В качестве примера применения suid можно привести права для команды passwd. Такой подход позволяет пользователям изменять собственные пароли.

```
$ ls -l /usr/bin/passwd
-rwsr-xr-x 1 root root ... /usr/bin/passwd
```

На тот факт, что для команды passwd установлен признак suid, указывает символ s, расположенный в той позиции, в которой располагается символ x для владельца файла. Владельцем passwd является пользователь root, но данная команда должна быть доступна для обычных пользователей, иначе они не смогут изменять свои пароли. С этой целью право x установлено для группы и для всех ос-

тальных пользователей. Однако этого недостаточно. Надо установить для passwd признак suid, чтобы любой пользователь, вызвавший эту команду, получал на время ее выполнения права root.

На заметку

Для обозначения установленного признака suid может использоваться как строчная буква s, так и прописная S. Строчная буква отображается в том случае, если перед установкой suid пользователь уже имел право на выполнение (x), а прописная буква S — если пользователь не имел такого права. Конечный результат будет один и тот же, но регистр символа дает дополнительную информацию об исходных установках.

Устанавливать и сбрасывать признак suid можно двумя способами: используя символьные либо числовые обозначения. Команда, в которой применены символьные обозначения, выглядит так, как показано ниже.

```
$ pwd
/home/scott/bin
$ ls -l
-rwxr-xr-- 1 scott admins ... backup_data
$ chmod u+s backup_data
$ ls -l
-rwsr-xr-- 1 scott admins ... backup_data
```

После завершения команды каждый член группы admins получает возможность запускать сценарий backup_data на выполнение от имени пользователя scott. Заметьте также, что пользователи, не принадлежащие группе admins, не могут сделать этого; они имеют право только на чтение файла. Если вам надо, чтобы любой пользователь мог запускать сценарии backup_data от имени пользователя scott, следует установить права -rwsr-xr-x.

Для того чтобы сбросить признак suid, надо вместо символа + ввести -.

```
$ ls -l
-rwsr-xr-- 1 scott admins ... backup_data
$ chmod u-s backup_data
$ ls -l
-rwxr-xr-- 1 scott admins ... backup_data
```

Использовать восьмеричные числа несколько сложнее, поскольку в данном случае вступают в действие дополнительные числовые признаки. До сих пор мы говорили о трехсимвольном наборе признаков, в котором первый символ представлял владельца файла, второй — группу и третий — остальных пользователей. Однако на самом деле в команде chmod предусмотрена четвертая цифра, расположенная слева от цифры, представляющей владельца файла. В большинстве случаев это цифра 0, поэтому указывать ее нет необходимости. Действительно, chmod 644 libby.jpg и chmod 0644 libby.jpg — одно и то же. Однако, когда возникает необходимость изменить признак suid (а также sgid или "sticky bit", которые будут рассмотрены в следующих разделах), эта цифра вступает в действие.

Для установки права suid используется значение 4, поэтому, если вы хотите изменить права для backup_data, используя числовые обозначения, команда будет выглядеть так, как показано ниже.

```
$ pwd
/home/scott/bin
$ ls -l
-rwxr-xr-- 1 scott admins ... backup_data
$ chmod 4754 backup_data
$ ls -l
-rwsr-xr-- 1 scott admins ... backup_data
```

Для сброса suid используется значение, равное 0.

```
$ ls -l
-rwsr-xr-- 1 scott admins ... backup_data
$ chmod 0754 backup_data
$ ls -l
-rwxr-xr-- 1 scott admins ... backup_data
```

На заметку

У вас, обычного пользователя, не наделенного специальными полномочиями, вряд ли возникнет необходимость изменять признак suid. Чаще всего этот признак устанавливается для программ, владельцем которых является пользователь root, однако всем остальным также желательно знать данный механизм, чтобы понимать, что происходит при вызове подобной программы.

Установка и сброс признака sgid

```
chmod g[+-]s
```

Признак sgid похож на suid, однако он применим не только к файлам, но и к каталогам. Для файлов признак sgid действует так же, как suid, за исключением того, что пользователь запускает файл на выполнение не с правами владельца, а с правами группы. Например, в вашей системе для файла crontab, вероятнее всего, установлен признак sgid, поэтому программы, заданные для cron, будут выполняться не с правами root, а с гораздо более ограниченными правами группы.

```
$ ls -l /usr/bin/crontab
-rwxr-sr-x 1 root crontab ... /usr/bin/crontab
```

В применении к каталогам sgid делает следующее: для каждого очередного файла, создаваемого в каталоге,

устанавливается принадлежность группе. Поясним сказанное на примере.

Предположим, в вашей системе зарегистрированы три пользователя, alice, bob и carol, которые являются членами группы admins. Для пользователя alice основной группой является alice. По тому же принципу формируются основные группы для пользователей bob и carol. Если alice создаст файл в каталоге, совместно используемом группой admins, то для него будут установлены владелец alice и группа alice. Это означает, что остальные члены группы admins не смогут записывать информацию в данный файл. Конечно же, alice может после создания файла выполнить команду chgrp admins document (или chown :admins document), однако поступать так каждый раз довольно утомительно.

Если же для каталога установлен признак sgid, каждый новый файл, созданный в этом каталоге, будет принадлежать пользователю, который создал файл, и той группе, которой принадлежит сам каталог, в данном случае admins. В результате alice, bob и carol смогут читать и редактировать любой файл, содержащийся в этом каталоге, причем для этого не потребуется прилагать дополнительные усилия.

Как и в случае suid, вы можете устанавливать признак sgid, используя либо символьные, либо числовые обозначения. Символьные обозначения применяются по тем же правилам, что и для признака suid, только вместо буквы u указывается g. Рассмотрим процесс установки признака sgid для каталога. Для файла этот признак задается точно так же.

```
$ ls -lF
drwxr-xr-x 11 scott admins ... bin/
$ chmod g+s bin
$ ls -lF
drwxr-Sr-x 11 scott admins ... bin/
```

На заметку

Для обозначения установленного признака sgid может использоваться как строчная буква s, так и прописная S. Строчная буква отображается в том случае, если перед установкой sgid группа уже имела право на выполнение (x), а заглавная S — если группа не имела такого права. Конечный результат будет один и тот же, но регистр символа дает дополнительную информацию об исходных установках.

Процесс удаления признака sgid — противоположный процессу его установки.

```
$ ls -lF
drwxr-Sr-x 11 scott admins ... bin/
$ chmod g-s bin
$ ls -lF
drwxr-xr-x 11 scott admins ... bin/
```

Если вы не читали предыдущий раздел, посвященный признаку suid, прочитайте его, иначе вам будет непонятно назначение первой из четырех цифр, указанных в команде chmod. Если для suid это была цифра 4, то для sgid используется значение 2.

```
$ ls -lF
drwxr-xr-x 11 scott admins ... bin/
$ chmod 2755 bin
$ ls -lF
drwxr-Sr-x 11 scott admins ... bin/
```

Удаляется признак sgid так же, как и suid: первая цифра задается равной 0.

```
$ ls -lF
drwxr-Sr-x 11 scott admins ... bin/
$ chmod 0755 bin
$ ls -lF
drwxr-xr-x 11 scott admins ... bin/
```

На заметку

Вы уже знаете, что происходит с новым файлом, который создается в каталоге, имеющем признак `sgid`. Однако необходимо заметить, что `sgid` влияет (причем по-разному) и на другие операции с файловой системой. Если вы скопируете файл в sgid-каталог посредством команды `cp`, он будет принадлежать той же группе, что и sgid-каталог. Если вы переместите в такой каталог файл, используя команду `mv`, он сохранит принадлежность прежней группе. И наконец, если вы создадите посредством команды `mkdir` новый каталог в составе каталога, имеющего признак `sgid`, он не унаследует группу sgid-каталога, но сам будет иметь признак `sgid`.

Установка и сброс признака "sticky bit"

```
chmod [+-]t
```

Чем замечателен признак "sticky bit", помимо непереводимого названия? На заре Unix, если данный признак был установлен для исполняемого файла, система знала, что файл используется часто, поэтому хранила его в области подкачки, обеспечивая быстрый и эффективный доступ к нему. Linux — более современная система, она игнорирует "sticky bit", установленный для файлов.

В настоящее время "sticky bit" применяется только к каталогам. Если данный признак установлен для каталога, то удалять и переименовывать содержащиеся в нем файлы может только владелец файлов или самого каталога. В обычных условиях разрешение записи в каталог также означает право изменять и переименовывать содержимое каталога. Признак "sticky bit" отменяет такое право. В качестве примера каталога, в который каждый желающий может записывать файл, можно привести `/tmp`. В этом же каталоге каждый файл и подкаталог защищен от остальных пользователей посредством признака "sticky bit".

```
$ ls -l /
drwxrwxrwt 12 root root ... tmp
[Данные сокращены для экономии места]
```

На заметку

Для обозначения установленного признака "sticky bit" может использоваться как строчная буква t, так и прописная T. Строчная буква отображается в том случае, если перед установкой "sticky bit" произвольный пользователь уже имел право на выполнение (x), а прописная T — если такого права у него не было. Конечный результат один и тот же, но регистр символа дает дополнительную информацию об исходных установках.

Подобно многим другим вариантам использования команды chmod, для установки признака "sticky bit" можно использовать либо буквенные, либо числовые обозначения.

```
$ ls -lF
drwxrwxr-x 2 scott family ... libby_pix/
$ chmod +t libby_pix
$ ls -lF
drwxrwxr-t 2 scott family ... libby_pix/
```

Однако процесс установки признака "sticky bit" имеет две существенные особенности. Во-первых, несмотря на то, что в предыдущих случаях необходимо было указать, для какой категории пользователей (u, g или o) задается соответствующее право, при установке "sticky bit" в этом нет необходимости. Достаточно указать +t. Во-вторых, символ t отображается в той позиции, в которой обычно выводится символ x для пользователей, не являющихся владельцами и не принадлежащих группе. Даже если запись в каталог разрешается только членам группы, все равно удалять или переименовывать файл имеет право только его владелец.

Удаление "sticky bit" происходит так, как вы, вероятно, и ожидаете.

```
$ ls -lF
drwxrwxr-t 2 scott family ... libby_pix/
$ chmod -t libby_pix
$ ls -lF
drwxrwxr-x 2 scott family ... libby_pix/
```

Установка признака "sticky bit" с помощью числовых обозначений предполагает указание четвертой цифры в составе параметра, передаваемого команде chmod. Если для suid это 4, а для sgid — 2, то для "sticky bit" используется значение 1.

```
$ ls -lF
drwxrwxr-x 2 scott family ... libby_pix/
$ chmod 1775 libby_pix
$ ls -lF
drwxrwxr-t 2 scott family ... libby_pix/
```

Значение 0, как и в предыдущих случаях, сбрасывает "sticky bit".

```
$ ls -lF
drwxrwxr-t 2 scott family ... libby_pix/
$ chmod 0775 libby_pix
$ ls -lF
drwxrwxr-x 2 scott family ... libby_pix/
```

Вы вряд ли будете часто использовать признак "sticky bit" на рабочей станции, но для серверов он очень удобен. Без него некоторые задачи, связанные с контролем доступа, было бы достаточно трудно решить.

Совет

Для ускорения работы можно одновременно устанавливать из командной строки признаки suid, sgid и "sticky bit". Подобно тому, как вы объединяете путем сложения значения 4 (чтение), 2 (запись) и 1 (выполнение), определяющие права пользователя, можно объединить suid, sgid и "sticky bit".

Числовое значение	Описание
0	Сбрасывает suid, sgid и "sticky bit"
1	Устанавливает "sticky bit"
2	Устанавливает sgid
3	Устанавливает "sticky bit" и sgid
4	Устанавливает suid
5	Устанавливает "sticky bit" и suid
6	Устанавливает sgid и suid
7	Устанавливает "sticky bit", sgid и suid

Заметьте также, что значение 0 одновременно удаляет suid, sgid и "sticky bit". Если вы использовали 0 для удаления suid, но хотите сохранить "sticky bit", вам надо повторно установить данный признак.

Выводы

Права доступа — чрезвычайно важный элемент системы защиты; по сути, он определяет работоспособность Linux. Зная принцип действия прав доступа, достаточно просто научиться устанавливать их в соответствии с собственными потребностями. Совместно команды chgrp (изменение принадлежности группе), chown (установка владельца, а также группы) и чрезвычайно мощная команда chmod формируют набор инструментов, позволяющие эффективно устанавливать права доступа.

8

Создание архивов и сжатие данных

Некоторые пользователи считают, что архивация файлов и их сжатие — одно и то же, однако эти понятия необходимо различать. Если вы берете десять файлов и, не изменяя их размеры, объединяете в один файл, вы создаете архив. Пусть каждый из десяти файлов имеет размер 100 Кбайт, тогда размер полученного в результате архива будет равен 1000 Кбайт. Если вы выполните сжатие тех же десяти файлов, то получите по-прежнему десять файлов, размеры которых, в зависимости от типа исходных данных, будут составлять от нескольких килобайт до 100 Кбайт.

На заметку

В принципе файл, получившийся в результате сжатия, может быть даже больше исходного файла. Если файл уже был сжат ранее, реального сжатия не произойдет, но в состав файла будет дополнительно включена служебная информация.

Все средства архивирования и сжатия, рассмотренные в данной главе, — `zip`, `gzip`, `bzip2` и `tar` — широко используются на практике, однако самым популярным все же является инструмент `zip`. Он не только реализует стандартный формат для системы Windows, но и широко применяется

в подавляющем большинстве операционных систем, поэтому zip-файл, созданный, например, в Linux, можно прочитать в Mac OS. Если ваш архив предназначен для некоторого пользователя и вы не знаете, с какой операционной системой он работает, лучше всего выбрать формат zip.

Программа gzip была разработана как проект с открытым исходным кодом и предназначалась на замену программы compress, которая использовалась ранее в системе Unix. Программа gzip имеется практически на каждом компьютере под управлением Unix, а также поддерживается в системе Mac OS X. В Windows она встречается значительно реже. Если вы собираетесь перемещать файлы между различными компьютерами, на которых установлена система Unix, можете смело использовать gzip.

Команда bzip2 сравнительно новая. Ее разработчики стремились получить результаты, лучшие по сравнению с gzip. Действительно, bzip2 создает файлы меньшего размера, но достигается это за счет увеличения времени обработки. В настоящее время быстродействие компьютеров настолько возросло, что большинство пользователей не заметят различия во времени при сжатии группы файлов посредством gzip и bzip2.

На заметку

В журнале *Linux Magazine* опубликована хорошая статья, в которой сравнивается несколько форматов сжатия. Статья доступна по адресу http://www.linux-mag.com/content/view/1678/43/.

Команды zip, gzip и bzip2 предназначены для сжатия данных (zip также может создавать архивы). Команда tar решает одну задачу — архивирование; именно для этого она разрабатывалась и используется длительное время. Она применяется в основном в системе Unix. Если вы копируете с сервера исходный код, он почти наверняка будет представлен в виде tar-архива.

Архивирование и сжатие файлов посредством программы zip

```
zip
```

Программа `zip` предназначена как для архивирования, так и для сжатия файлов. Она удобна, например, для передачи нескольких файлов в составе почтового сообщения, формирования резервных копий или для хранения на диске данных, которые используются достаточно редко. Пользоваться программой `zip` очень просто. Предположим, что вы хотите передать изображение в формате TIFF по электронной почте. Изображение TIFF хранится в несжатом виде, поэтому файл имеет большой размер. Обработав такой файл программой `zip`, вы получите файл меньшего размера, который можно присоединить к электронному письму.

На заметку

Для экономии места из результатов, полученных по команде `ls -l`, представлены только те, которые имеют отношение к рассматриваемому примеру.

```
$ ls -lh
-rw-r--r-- scott scott 1006K young_edgar_scott.tif
$ zip grandpa.zip young_edgar_scott.tif
  adding: young_edgar_scott.tif (deflated 19%)
$ ls -lh
-rw-r--r-- scott scott 1006K young_edgar_scott.tif
-rw-r--r-- scott scott 819K grandpa.zip
```

В данном примере мы сэкономили около 200 Кбайт или, по сообщению `zip`, 19% объема. Не так плохо. То же самое можно сделать для нескольких изображений.

```
$ ls -l
-rw-r--r--  scott  scott  251980
  edgar_intl_shoe.tif
-rw-r--r--  scott  scott  1130922
  edgar_baby.tif
-rw-r--r--  scott  scott  1029224
  young_edgar_scott.tif
$ zip grandpa.zip edgar_intl_shoe.tif
  edgar_baby.tif young_edgar_scott.tif
adding: edgar_intl_shoe.tif (deflated 4%)
adding: edgar_baby.tif (deflated 12%)
adding: young_edgar_scott.tif (deflated 19%)
$ ls -l
-rw-r--r--  scott  scott  251980
  edgar_intl_shoe.tif
-rw-r--r--  scott  scott  1130922  edgar_baby.tif
-rw-r--r--  scott  scott  2074296  grandpa.zip
-rw-r--r--  scott  scott  1029224
  young_edgar_scott.tif
```

Указывать программе `zip` отдельные файлы — не лучшее решение. Для трех файлов это приемлемо. Пользователь, выполнив команду `unzip grandpa.zip`, получит всего три файла. Если же в архиве содержится пятьдесят файлов, то после разархивирования файлы, извлеченные из архива, смешаются с теми файлами, которые уже находились в каталоге, и разобраться с ними будет непросто. Гораздо лучше поместить эти пятьдесят файлов в отдельный каталог, который пользователь и получит в результате разархивирования.

```
$ ls -lF
drwxr-xr-x  scott  scott  edgar_scott/
$ zip grandpa.zip edgar_scott
adding: edgar_scott/ (stored 0%)
```

```
adding: edgar_scott/edgar_baby.tif (deflated 12%)
adding: edgar_scott/young_edgar_scott.tif
  (deflated 19%)
adding: edgar_scott/edgar_intl_shoe.tif
  (deflated 4%)
$ ls -lF
drwxr-xr-x scott scott 160 edgar_scott/
-rw-r--r--  scott  scott  2074502  grandpa.zip
```

Независимо от того, архивируется ли файл, несколько файлов или каталог, выполняемые действия одинаковы: задается команда `zip`, за ней указывается имя zip-архива, а после него — объект или объекты, которые необходимо добавить к архиву.

Повышение уровня сжатия с помощью программы zip

```
-[0-9]
```

Программе `zip` можно задавать различные уровни сжатия данных. Эти уровни обозначаются числами от 0 до 9, причем 0 означает отсутствие сжатия (в этом случае `zip` работает подобно программе `tar`, которую мы рассмотрим далее в этой главе), 1 задает большую производительность и малую степень сжатия, а 9 соответствует максимальной степени сжатия, достигаемой за счет снижения скорости обработки данных. По умолчанию принимается уровень 6, однако быстродействие современных компьютеров велико и можно всегда использовать уровень 9.

Предположим, вас заинтересовало произведение Мелвилла "Моби Дик" и вы хотите собрать тексты, которые помогут вам понять эту книгу: само произведение "Моби Дик", "Утерянный рай" Мильтона и текст из Библии. Сравним результаты при различных уровнях сжатия.

```
$ ls -l
-rw-r--r--  scott  scott   102519  job.txt
-rw-r--r--  scott  scott  1236574  moby-dick.txt
-rw-r--r--  scott  scott   508925
  paradise_lost.txt
$ zip -0 moby.zip *.txt
adding: job.txt (stored 0%)
adding: moby-dick.txt (stored 0%)
adding: paradise_lost.txt (stored 0%)
$ ls -l
-rw-r--r--  scott  scott   102519  job.txt
-rw-r--r--  scott  scott  1236574  moby-dick.txt
-rw-r--r--  scott  scott  1848444  moby.zip
-rw-r--r--  scott  scott   508925
  paradise_lost.txt
$ zip -1 moby.zip *txt
updating: job.txt (deflated 58%)
updating: moby-dick.txt (deflated 54%)
updating: paradise_lost.txt (deflated 50%)
$ ls -l
-rw-r--r--  scott  scott   102519  job.txt
-rw-r--r--  scott  scott  1236574  moby-dick.txt
-rw-r--r--  scott  scott   869946  moby.zip
-rw-r--r--  scott  scott   508925
  paradise_lost.txt
$ zip -9 moby.zip *txt
updating: job.txt (deflated 65%)
updating: moby-dick.txt (deflated 61%)
updating: paradise_lost.txt (deflated 56%)
$ ls -l
-rw-r--r--  scott  scott   102519  job.txt
-rw-r--r--  scott  scott  1236574  moby-dick.txt
-rw-r--r--  scott  scott   747730  moby.zip
-rw-r--r--  scott  scott   508925
  paradise_lost.txt
```

В виде таблицы результаты можно представить следующим образом.

Книга	zip -0	zip -1	zip -9
Моби-Дик	0%	54%	61%
Утерянный рай	0%	50%	56%
Текст из Библии	0%	58%	65%
Общий размер (в байтах)	1848444	869946	747730

Степень сжатия существенно зависит от типа файла (тексты обычно сжимаются очень хорошо) и размеров исходных файлов. Очевидно, что лучше использовать уровень сжатия, равный 9. Другое решение целесообразно, если ваш компьютер имеет очень низкое быстродействие, либо если сами вы крайне нетерпеливы.

На заметку

Целесообразно создать в файле `.bashrc` псевдоним для команды `zip`. Соответствующая запись будет выглядеть следующим образом:

```
alias zip='zip -9'
```

Теперь при каждом вызове `zip` будет автоматически задаваться максимальный уровень сжатия.

Защита zip-архивов паролем

```
-P
-e
```

Программа `zip` позволяет защищать паролем создаваемые zip-архивы. Для этой цели предусмотрена опция `-P`. Однако использовать ее не рекомендуется, так как она чрезвычайно уязвима с точки зрения безопасности. Это хорошо

видно из следующего примера (в данном случае задан пароль 12345678):

```
$ zip -P 12345678 moby.zip *.txt
```

Поскольку пароль приходится задавать в командной строке, каждый желающий, просматривая предысторию работы с оболочкой (а сделать это очень просто), увидит его. Итак, не используйте опцию `-P`!

Лучше применять для этой цели опцию `-e`, которая шифрует содержимое zip-файла и требует ввода пароля. В этом случае пароль не задается в командной строке, а вводится в ответ на приглашение. В результате он не попадает в список предыстории.

```
$ zip -e moby.zip *.txt
Enter password:
Verify password:
adding: job.txt (deflated 65%)
adding: moby-dick.txt (deflated 61%)
adding: paradise_lost.txt (deflated 56%)
```

В этом списке сохранится только вызов команды `zip -e moby.zip *.txt`. Реальный пароль останется в секрете.

Внимание!

Уровень безопасности, обеспечиваемый программой `zip`, невысок. В глобальной сети можно найти немало инструментов, позволяющих быстро взломать zip-архив, защищенный паролем. Архив с паролем можно сравнить с письмом в конверте. Для обычного человека оно остается закрытым, но, попав в руки злоумышленника, оно, конечно же, будет прочитано.

Кроме того, версия программы `zip`, поставляемая в составе некоторых дистрибутивных пакетов Linux, не поддерживает шифрование. В этом случае вы получите сообщение об ошибке `encryption not supported`. Единственным решением остается перекомпиляция программы `zip` из исходного текста.

Разархивирование файлов

```
unzip
```

Разархивирование zip-архива осуществляется очень просто. Если для того, чтобы создать архив, используется команда zip, то для разархивирования служит команда unzip.

```
$ unzip moby.zip
Archive: moby.zip
inflating: job.txt
inflating: moby-dick.txt
inflating: paradise_lost.txt
```

Команда unzip подробно сообщает о всех выполняемых действиях. Чтобы получить еще более детальную информацию, следует задать опцию -v (сокращение от слова verbose).

```
unzip -v moby.zip
Archive: moby.zip
Length   Method   Size  Ratio CRC-32   Name
-------  ------ ------- ----- ------   ----
102519   Defl:X  35747   65% fabf86c9 job.txt
1236574  Defl:X 487553   61% 34a8cc3a moby-dick.txt
508925   Defl:X 224004   56% 6abe1d0f
  paradise_lost.t
-------          ------  ---          -------
1848018          747304  60%          3 files
```

Как видите, в этом случае программа сообщает о методе, использованном для сжатия файлов, размере сжатого файла в процентах от исходного размера и значении CRC (оно используется для коррекции ошибок).

Получение списка файлов для разархивирования

```
-l
```

Может случиться так, что вы не вспомните, какие файлы содержатся в том или ином архиве. А возможно, вы не уверены в том, что нужный вам файл находится именно в том архиве, с которым вы собираетесь работать в данный момент. Вывести содержимое zip-файла позволяет опция `-l` (сокращение от слова `list`).

```
$ unzip -l moby.zip
Archive: moby.zip
  Length      Date    Time    Name
---------  ---------- -----   ----
        0  01-26-06 18:40 bible/
   207254  01-26-06 18:40 bible/genesis.txt
   102519  01-26-06 18:19 bible/job.txt
  1236574  01-26-06 18:19 moby-dick.txt
   508925  01-26-06 18:19 paradise_lost.txt
---------                  -------
  2055272                  5 files
```

В результате видно, что в архиве `moby.zip` содержатся два файла, `moby-dick.txt` и `paradise_lost.txt`, а также каталог `bible`, который, в свою очередь, содержит два файла: `genesis.txt` и `job.txt`. Теперь вы точно знаете, что произойдет при разархивировании архива `moby.zip`, и не получится так, что, применив программу `unzip` к неизвестному архиву, вы запишете в текущий каталог сотню отдельных файлов.

Проверка файлов, предназначенных для разархивирования

```
-t
```

Файл архива может быть поврежден. Хуже всего, если вы разархивируете файлы и удалите архив, а лишь потом обнаружите, что некоторые или даже все файлы невозможно открыть. Гораздо лучше проверить архив до разархивации. Для этой цели предусмотрена опция `-t` (сокращение от слова `test`).

```
$ unzip -t moby.zip
Archive: moby.zip
    testing: bible/                  OK
    testing: bible/genesis.txt       OK
    testing: bible/job.txt           OK
    testing: moby-dick.txt           OK
    testing: paradise_lost.txt       OK
No errors detected in compressed data of moby.zip.
```

Желательно каждый раз при работе с zip-архивом использовать опцию `-t`. Несмотря на то что это потребует дополнительного времени, в конце концов вы компенсируете затраты за счет повышения надежности своей работы.

Сжатие файлов посредством программы gzip

```
gzip
```

Использовать программу `gzip` несколько проще, чем `zip`. Работая с программой `zip`, вы должны указать имя создаваемого zip-файла. Программа `gzip` требует лишь, чтобы вы ввели имя команды и имя файла, подлежащего сжатию.

```
$ ls -l
-rw-r--r--  scott  scott  508925  paradise_lost.txt
$ gzip paradise_lost.txt
$ ls -l
-rw-r--r--  scott  scott  224425
  paradise_lost.txt.gz
```

Необходимо принимать во внимание существенное различие между программами `zip` и `gzip`. Если вы обрабатываете файл с помощью `zip`, он остается в неприкосновенности, т.е. вы получаете и исходный файл, и вновь созданный zip-архив. Программа `gzip` удаляет исходный файл и оставляет только сжатый файл.

Если вы хотите, чтобы после обработки `gzip` исходный файл не удалялся, используйте опцию `-c` (или `--stdout` или `--to-stdout`), которая выводит результаты работы программы `gzip` в стандартный выходной поток, и перенаправьте вывод в другой файл. Если вы зададите опцию `-c` и забудете перенаправить вывод, то получите на экране странный набор символов, подобный приведенному ниже.

```
$ gzip -c paradise_lost.txt
w
  I
?l?,(3э??_?i +??M?S3?t1*f%eY??'[q??
D??}d]C%g?       R?@,r?e-trB3+3/??|*??OD?-s
BAqn??,Y8*#"]]RU
*b?U\????G???'t(-??x?Yz3-?o'~cnS??K
_?c?
```

Перенаправление в файл выглядит следующим образом:

```
$ ls -l
-rw-r--r-- 1 scott scott 508925 paradise_lost.txt
$ gzip -c paradise_lost.txt
  > paradise_lost.txt.gz
```

```
$ ls -l
-rw-r--r-- 1  scott  scott  497K
  paradise_lost.txt
-rw-r--r-- 1  scott  scott  220K
  paradise_lost.txt.gz
```

Теперь результаты значительно лучше. В вашем распоряжении есть и исходный файл, и его сжатая версия.

Совет

Если вы случайно задали опцию -c и не перенаправили вывод, нажмите несколько раз комбинацию клавиш <Ctrl+C>, чтобы прервать работу программы `gzip`.

Рекурсивная обработка файлов посредством программы gzip

```
-r
```

Если вы хотите обработать командой `gzip` несколько файлов, содержащихся в одном каталоге, вам надо использовать символы групповых операций. Однако результаты могут отличаться от тех, которые вы ожидаете. Это демонстрирует следующий пример:

```
$ ls -F
bible/ moby-dick.txt paradise_lost.txt
$ ls -l *
-rw-r--r--  scott  scott  1236574  moby-dick.txt
-rw-r--r--  scott  scott   508925
  paradise_lost.txt

bible:
-rw-r--r--  scott  scott  207254  genesis.txt
-rw-r--r--  scott  scott  102519  job.txt
$ gzip *
```

```
gzip: bible is a directory -- ignored
$ ls -l *
-rw-r--r--  scott  scott  489609  moby-dick.txt.gz
-rw-r--r--  scott  scott  224425
  paradise_lost.txt.gz

bible:
-rw-r--r--  scott  scott  207254  genesis.txt
-rw-r--r--  scott  scott  102519  job.txt
```

Заметьте, что файлы в каталоге `bible` остались необработанными; по умолчанию `gzip` не затрагивает содержимого подкаталогов. Для того чтобы задать рекурсивную обработку, надо использовать не только символ групповой операции, но и опцию `-r` (или `--recursive`).

```
$ ls -F
bible/ moby-dick.txt paradise_lost.txt
$ ls -l *
-rw-r--r--  scott  scott  1236574  moby-dick.txt
-rw-r--r--  scott  scott   508925
  paradise_lost.txt

bible:
-rw-r--r--  scott  scott  207254  genesis.txt
-rw-r--r--  scott  scott  102519  job.txt
$ gzip -r *
$ ls -l *
-rw-r--r--  scott  scott  489609
  moby-dick.txt.gz
-rw-r--r--  scott  scott  224425
  paradise_lost.txt.gz

bible:
-rw-r--r--  scott  scott  62114  genesis.txt.gz
-rw-r--r--  scott  scott  35984  job.txt.gz
```

На этот раз сжатию подверглись все файлы, даже те, которые содержались в подкаталоге. Однако сжатый вариант каждого файла сохраняется независимо от других. В отличие от `zip`, команда `gzip` не объединяет все файлы в один большой файл. Для того чтобы сделать это, надо использовать команду `tar`, которую мы рассмотрим далее в этой главе.

Повышение уровня сжатия с помощью программы gzip

```
-[0-9]
```

Как и `zip`, программа `gzip` позволяет задавать уровень сжатия. Эти уровни обозначаются числами от 0 до 9, причем 0 означает отсутствие сжатия (в этом случае `gzip` работает подобно программе `tar`, которую мы рассмотрим ниже), 1 задает большую скорость обработки и малую степень сжатия, а 9 соответствует максимальной степени сжатия, достигаемой за счет сравнительно медленной обработки данных. По умолчанию принимается уровень 6, однако быстродействие современных компьютеров велико, поэтому можно всегда использовать уровень 9.

```
$ ls -l
-rw-r--r--  scott  scott  1236574  moby-dick.txt
$ gzip -c -1 moby-dick.txt > moby-dick.txt.gz
$ ls -l
-rw-r--r--  scott  scott  1236574  moby-dick.txt
-rw-r--r--  scott  scott   571005  moby-dick.txt.gz
$ gzip -c -9 moby-dick.txt > moby-dick.txt.gz
$ ls -l
-rw-r--r--  scott  scott  1236574  moby-dick.txt
-rw-r--r--  scott  scott   487585  moby-dick.txt.gz
```

Как вы помните, задав опцию -c и перенаправив вывод в файл, можно предотвратить удаление исходного файла.

На заметку

Целесообразно создать в файле .bashrc псевдоним для команды gzip. Соответствующая запись будет выглядеть следующим образом:

```
alias gzip='gzip -9'
```

Теперь при каждом вызове gzip будет автоматически задаваться максимальный уровень сжатия.

Распаковка файлов, сжатых с помощью программы gzip

```
gunzip
```

Для распаковки файлов, сжатых посредством gzip, используется команда gunzip.

```
$ ls -l
-rw-r--r--  scott  scott  224425
  paradise_lost.txt.gz
$ gunzip paradise_lost.txt.gz
$ ls -l
-rw-r--r--  scott  scott  508925
  paradise_lost.txt
```

Подобно тому, как программа gzip удаляет исходный файл, оставляя лишь результат его обработки, gunzip удаляет файл .gz, предоставляя пользователю только распакованный файл. Если вы хотите сохранить оба файла, вам надо задать опцию -c (или --stdout, или --to-stdout) и перенаправить вывод в файл.

```
$ ls -l
-rw-r--r--  scott  scott  224425
  paradise_lost.txt.gz
$ gunzip -c paradise_lost.txt.gz
  > paradise_lost.txt
$ ls -l
-rw-r--r--  scott  scott  508925
  paradise_lost.txt
-rw-r--r--  scott  scott  224425
  paradise_lost.txt.gz
```

Опция `-c` удобна в случае, если вам надо оставить у себя распакованный файл и передать кому-нибудь сжатый. Конечно, вы снова сможете использовать `gzip`, но зачем выполнять лишнюю работу?

На заметку

Если вам по каким-то причинам не нравится команда `gunzip`, вы можете использовать вместо нее команду `gzip` с опцией `-d` (или `--decompress`, или `--uncompress`).

Проверка файлов, предназначенных для распаковки с помощью программы gunzip

```
-t
```

Перед тем как распаковывать файл, желательно проверить, не поврежден ли он. Для этого можно использовать опцию `-t` (или `--test`).

```
$ gzip -t paradise_lost.txt.gz
$
```

Если архив в порядке, `gzip` не выведет никакого сообщения. При наличии проблем программа оповестит вас

об этом. Возможно, поведение программы при отсутствии проблем покажется кому-то странным, но таков общий принцип работы утилит Unix: информация выводится только в том случае, когда от пользователя требуются специальные действия.

Сжатие файлов посредством программы bzip2

```
bzip2
```

Работать с программой bzip2 очень легко. Если вы не испытываете затруднений при использовании программы gzip, то обязательно справитесь и с bzip2. Разработчики этой утилиты позаботились о том, чтобы набор ее опций и поведение были максимально похожи на gzip.

```
$ ls -l
-rw-r--r--  scott  scott  1236574  moby-dick.txt
$ bzip2 moby-dick.txt
$ ls -l
-rw-r--r--  scott  scott   367248
  moby-dick.txt.bz2
```

Подобно gzip, программа bzip2 оставляет только файл .bz2, а исходный файл удаляет. Для того чтобы сохранить его, надо задать опцию -c (или --stdout) и переназначить вывод в файл, имя которого оканчивается символами .bz2.

```
$ ls -l
-rw-r--r--  scott  scott  1236574  moby-dick.txt
$ bzip2 -c moby-dick.txt > moby-dick.txt.bz2
$ ls -l
-rw-r--r--  scott  scott  1236574  moby-dick.txt
-rw-r--r--  scott  scott   367248
  moby-dick.txt.bz2
```

Если вы прочитали разделы, посвященные программе gzip, то заметите, что gzip и bzip2 ведут себя одинаково.

Повышение уровня сжатия с помощью программы bzip2

```
-[0-9]
```

Подобно zip и gzip, программа bzip2 позволяет задавать уровень сжатия. Уровни обозначаются числами от 0 до 9, причем 0 означает отсутствие сжатия (в этом случае bzip2 работает подобно программе tar), 1 задает большую скорость обработки и малую степень сжатия, а 9 соответствует максимальной степени сжатия, достигаемой за счет снижения скорости обработки данных. По умолчанию принимается уровень 6, однако быстродействие современных компьютеров велико, поэтому можно всегда использовать уровень 9.

```
$ ls -l
-rw-r--r--  scott  scott  1236574  moby-dick.txt
$ bzip2 -c -1 moby-dick.txt > moby-dick.txt.bz2
$ ls -l
-rw-r--r--  scott  scott  1236574  moby-dick.txt
-rw-r--r--  scott  scott   424084
  moby-dick.txt.bz2
$ bzip2 -c -9 moby-dick.txt > moby-dick.txt.bz2
$ ls -l
-rw-r--r--  scott  scott  1236574  moby-dick.txt
-rw-r--r--  scott  scott   367248
  moby-dick.txt.bz2
```

Как легко заметить в последнем примере, при переходе от уровня 1 к уровню 9 размер файла сократился с 424 до 367 Кбайт. Конечно же, различие существенное! Обратите

также внимание на различие размеров файлов, полученных с помощью программ gzip и bzip2. При использовании уровня 9 программа gzip сжала файл moby-dick.txt до 488 Кбайт, а bzip2 — до 367 Кбайт. Программа bzip2 работает медленнее, чем gzip, но на современных машинах дополнительная задержка составляет две-три секунды, что вполне приемлемо.

На заметку

Целесообразно создать в файле .bashrc псевдоним для команды bzip2. Соответствующая запись будет выглядеть следующим образом:

```
alias bzip2='bzip2 -9'
```

Теперь при каждом вызове bzip2 будет автоматически задаваться максимальный уровень сжатия.

Распаковка файлов, сжатых с помощью программы bzip2

```
bunzip2
```

Если bzip2 разработана так, чтобы быть максимально похожей на gzip, то bunzip2 ведет себя практически так же, как и gunzip.

```
$ ls -l
-rw-r--r--  scott  scott   367248
  moby-dick.txt.bz2
$ bunzip2 moby-dick.txt.bz2
$ ls -l
-rw-r--r--  scott  scott  1236574  moby-dick.txt
```

Обе команды удаляют сжатый файл, оставляя только результат его обработки. Если вы хотите сохранить оба

файла, вам надо указать опцию -c (или --stdout, или --to-stdout) и перенаправить вывод в файл, который вы собираетесь создать.

```
$ ls -l
-rw-r--r--  scott  scott   367248
  moby-dick.txt.bz2
$ bunzip2 -c moby-dick.txt.bz2 > moby-dick.txt
$ ls -l
-rw-r--r--  scott  scott  1236574  moby-dick.txt
-rw-r--r--  scott  scott   367248
  moby-dick.txt.bz2
```

Поскольку программы bzip2 и bunzip2 очень похожи на своих предшественниц, gzip и gunzip, изучить их очень легко. Разработчики поступили очень мудро.

На заметку

Если вам по каким-то причинам не нравится команда bunzip2, вы можете использовать для распаковки файлов команду bzip2 с опцией -d (или --decompress, или --uncompress).

Проверка файлов, предназначенных для разархивирования с помощью программы bunzip2

```
-t
```

Перед тем как распаковывать файл, желательно проверить, не поврежден ли он. Для этого можно использовать опцию -t (или --test).

```
$ bunzip2 -t paradise_lost.txt.gz
$
```

Подобно программе gunzip, если ошибки в архиве не обнаружены, bunzip2 не отображает никаких сведений. При наличии проблем программа оповестит вас об этом.

Архивирование файлов с помощью программы tar

```
-cf
```

Как уже упоминалось ранее в этой главе, программа tar не осуществляет сжатие, она лишь создает архивы. Для сжатия сформированных ею архивов tar использует другие программы, например gzip или bzip2. Если вы не собираетесь сжимать архив, вам надо использовать при его создании основные опции -c (или --create), которая сообщает tar о том, что создается архив, и -f (или --file), посредством которой задается имя архивного файла.

```
$ ls -l
scott  scott   102519  job.txt
scott  scott  1236574  moby-dick.txt
scott  scott   508925  paradise_lost.txt
$ tar -cf moby.tar *.txt
$ ls -l
scott  scott   102519  job.txt
scott  scott  1236574  moby-dick.txt
scott  scott  1853440  moby.tar
scott  scott   508925  paradise_lost.txt
```

Работая с программой tar, необходимо учитывать две особенности. Во-первых, объем архива больше, чем суммарный объем входящих в него файлов. Например, общий размер файлов job.txt, moby-dick.txt и paradise_lost.txt составляет 1848018 байт. Сравнив это значение с размером файла moby.tar, вы увидите, что архив на 5422 байта

больше. Не забывайте, что tar выполняет архивацию, а не сжатие, поэтому к суммарному размеру файлов, помещенных в архив, добавляется служебная информация. Во-вторых, заметьте, что tar, в отличие от gzip и bzip2, не удаляет исходные файлы. Это неудивительно, если учесть, что первоначально программа tar разрабатывалась как инструмент для создания резервных копий.

Способность объединять большое количество файлов и каталогов в один большой файл — основное преимущество программы tar по сравнению с программами gzip и bzip2.

```
$ ls -lF
drwxr-xr-x  scott  scott   168  moby-dick/
$ ls -l moby-dick/*
scott  scott   102519  moby-dick/job.txt
scott  scott  1236574  moby-dick/moby-dick.txt
scott  scott   508925  moby-dick/paradise_lost.txt

moby-dick/bible:
scott  scott  207254  genesis.txt
scott  scott  102519  job.txt
$ tar -cf moby.tar moby-dick/
$ ls -lF
scott  scott     168  moby-dick/
scott  scott 2170880  moby.tar
```

Программа tar постоянно используется с ранних версий Unix и до настоящего времени. Причину ее популярности понять нетрудно: она очень удобна. Однако данная команда становится еще удобнее, если использовать ее совместно с инструментом сжатия. Этот вопрос будет рассмотрен в следующем разделе.

Создание архивов и сжатие файлов посредством программ tar и gzip

```
-zcvf
```

Если вы помните материал разделов, посвященных программам gzip и bzip2, вы легко поймете суть проблемы. Предположим, что вам надо сжать каталог, содержащий сто файлов, распределенных по различным подкаталогам. Если вы используете для этой цели программу gzip или bzip2 с опцией -r, то получите сто отдельных сжатых файлов, находящихся в тех же подкаталогах. Такой результат вряд ли устраивает вас. Если, например, все эти данные надо переслать по почте, как присоединить к письму сто файлов?

На помощь приходит программа tar. Сначала надо использовать tar для архивирования каталога и его содержимого (сто файлов в различных подкаталогах), а затем сжать полученный архив с помощью программ gzip или bzip2. Поскольку gzip используется совместно с tar чаще, чем другие программы, мы сосредоточим внимание именно на ней.

Задачу архивации большого количества файлов можно решить следующим образом:

```
$ ls -l moby-dick/*
scott  scott   102519  moby-dick/job.txt
scott  scott  1236574  moby-dick/moby-dick.txt
scott  scott   508925  moby-dick/paradise_lost.txt

moby-dick/bible:
scott  scott  207254  genesis.txt
scott  scott  102519  job.txt
$ tar -cf moby.tar moby-dick/ | gzip -c >
moby.tar.gz
$ ls -l
scott  scott  168  moby-dick/
scott  scott   20  moby.tar.gz
```

Такой подход даст нужный результат, но при этом выражение в командной строке получается достаточно длинным. Существует более простой способ. Надо использовать следующие опции tar: -z (или --gzip), которая вызывает gzip и применяет ее к данным, сгенерированным tar, и -v (или --verbose), которая оповестит вас о выполняемых действиях. Заметьте, что указывать опцию -v не обязательно, но предоставляемые ею сведения позволяют убедиться в том, что все идет так, как вы предполагали.

```
$ ls -l moby-dick/*
scott  scott   102519  moby-dick/job.txt
scott  scott  1236574  moby-dick/moby-dick.txt
scott  scott   508925  moby-dick/paradise_lost.txt

moby-dick/bible:
scott  scott   207254  genesis.txt
scott  scott   102519  job.txt
$ tar -zcvf moby.tar.gz moby-dick/
moby-dick/
moby-dick/job.txt
moby-dick/bible/
moby-dick/bible/genesis.txt
moby-dick/bible/job.txt
moby-dick/moby-dick.txt
moby-dick/paradise_lost.txt
$ ls -l
scott  scott     168  moby-dick
scott  scott  846049  moby.tar.gz
```

Обычно файл, полученный в результате выполнения программ tar и gzip, оканчивается символами .tar.gz, однако при желании вы можете задать суффикс .tgz или .tar.gzip.

На заметку

Вместо `gzip` совместно с программой `tar` можно использовать `bzip2`. При этом команда будет выглядеть так (обратите внимание на опцию `-j`, которая задает вызов `bzip2`):

```
$ tar -jcvf moby.tar.bz2 moby-dick/
```

В данном случае имя файла будет оканчиваться символами `.tar.bz2`, но вы можете также задать окончания `.tar.bzip2`, `.tbz2` или `.tbz`. Применяя `bzip2` совместно с `tar`, желательно воздерживаться от использования суффикса `.tbz`, поскольку он очень похож на `.tgz`, генерируемый при работе с программой `gzip`.

Проверка файлов, предназначенных для распаковки и разархивирования

```
-zvtf
```

Перед тем как разархивировать файлы, содержащиеся в tar-архиве (независимо от того, сжат он или нет), желательно убедиться в том, что они не были повреждены. Во-первых, зная, что архив не в порядке, вы не будете тратить время, пытаясь организовать работу с содержащимися в нем файлами. Во-вторых, это позволит выявить ситуацию, когда пользователь, создававший архив, вместо каталога, содержащего сто файлов, включил в него сто отдельных файлов. При разархивировании подобного архива необходимо принять меры, чтобы находящиеся в нем файлы не смешались с файлами, расположенными в вашем текущем каталоге.

Для проверки tar-архива (предположим, что он был также сжат) используется опция `-t` (или `--test`).

```
$ tar -zvtf moby.tar.gz
scott/scott         0  moby-dick/
scott/scott    102519  moby-dick/job.txt
```

```
scott/scott        0  moby-dick/bible/
scott/scott   207254  moby-dick/bible/genesis.txt
scott/scott   102519  moby-dick/bible/job.txt
scott/scott  1236574  moby-dick/moby-dick.txt
scott/scott   508925  moby-dick/paradise_lost.txt
```

В результате вы получите информацию о правах доступа, владельце, размере и времени создания каждого файла. Поскольку каждая строка начинается с `moby-dick/`, вы можете сделать вывод, что в архиве находится каталог, содержащий файлы и каталоги.

При вводе команды убедитесь, что опция `-f` расположена последней, так как после нее указывается имя файла `.tar.gz`. Если вы зададите `-f` в середине последовательности опций, программа `tar` сообщит об ошибке.

```
$ tar -zvft moby.tar.gz
tar: You must specify one of the  -Acdtrux'
  options
Try  tar --help' or  tar --usage' for
  more information.
```

После того как вы убедились, что ваш файл `.tar.gz` не был поврежден, надо извлечь его содержимое. Как это сделать, вы узнаете в следующем разделе.

На заметку

Если вы проверяете tar-архив, сжатый посредством программы `bzip2`, вам надо использовать следующую команду:

```
$ tar -jvtf moby.tar.bz2
```

Распаковка и разархивирование файлов

```
-zxvf
```

Для того чтобы создать файл `.tar.gz`, мы использовали набор опций `-zcvf`. Для распаковки и разархивирования полученного файла необходимо лишь заменить опцию `-c` (или `--create`) опцией `-x` (или `--extract`).

```
$ ls -l
rsgranne rsgranne 846049 moby.tar.gz
$ tar -zxvf moby.tar.gz
moby-dick/
moby-dick/job.txt
moby-dick/bible/
moby-dick/bible/genesis.txt
moby-dick/bible/job.txt
moby-dick/moby-dick.txt
moby-dick/paradise_lost.txt
$ ls -l
rsgranne  rsgranne     168  moby-dick
rsgranne  rsgranne  846049  moby.tar.gz
```

Перед разархивированием файла его следует проверить. Это означает, что последовательность выполняемых команд должна выглядеть так:

```
$ tar -zvtf moby.tar.gz
$ tar -zxvf moby.tar.gz
```

На заметку

Если вы разархивируете tar-архив, сжатый посредством программы `bzip2`, вам надо использовать следующую команду:

```
$ tar -jxvf moby.tar.bz2
```

Выводы

В то время, когда скорость модемов была низкой, а объем жестких дисков маленьким, архивирование и сжатие файлов имело огромное значение. Сейчас это лишь вопрос удобства в работе, однако большинство пользователей постоянно выполняют подобные действия. Например, если вы когда-либо копировали исходный код, предназначенный для компиляции, он почти наверняка был расположен в сжатом tar-архиве. В будущем, вероятно, будут часто встречаться файлы, имена которых оканчиваются символами `.tar.bz2`. Если же вы обмениваетесь файлами с пользователями Windows, то, скорее всего, применяете для этой цели zip-архивы. Знать, как работают средства архивирования и сжатия, и уметь пользоваться ими гораздо важнее, чем может показаться на первый взгляд.

9

Поиск данных

С каждым годом объем жестких дисков возрастает, а цена на них снижается. Появление новых технических средств — цифровых фотоаппаратов, видеокамер, mp3-плейеров — приводит к генерации большого объема данных, которыми заполняется дополнительное дисковое пространство. Увеличение объема доступной информации порождает новые проблемы: найти нужные данные оказывается все труднее. Непросто отыскать фотоснимок среди 10000 изображений или текстовый файл среди 600 других документов. К счастью, в системе Linux есть мощные средства поиска, позволяющие быстро и эффективно найти требуемую информацию.

Поиск в базе имен файлов

```
locate
```

Бывает ли у вас такое, что вы знаете имя файла или часть имени, но не представляете, в каком месте файловой системы он находится? Решить эту проблему вам поможет команда `locate`. Она ищет файлы и каталоги по заданному имени. Результаты поиска выводятся на терминал.

```
$ locate haggard
.../txt/rider_haggard
.../txt/rider_haggard/Queen_of_the_Dawn.txt
.../txt/rider_haggard/Allan_and_the_Ice-Gods.txt
.../txt/rider_haggard/Heu-Heu_or_The_Monster.txt
```

Результаты отображаются быстро, однако программа `locate` не выполняет реального поиска. Вместо этого она просматривает базу данных имен файлов, которая ежедневно автоматически обновляется (подробнее этот вопрос будет рассмотрен далее в данной главе). Поскольку `locate` ищет информацию в предварительно созданной базе, она выполняет поставленную перед ней задачу практически мгновенно.

На вашем компьютере, возможно, вместо `locate` установлена программа `slocate`. Команда `slocate` (secure locate) не предпринимает попыток поиска в тех каталогах, которые пользователь, вызвавший ее, не имеет права просматривать (например, если вы не являетесь пользователем `root`, вам недоступно содержимое каталога `/root`). До появления `slocate` программа `locate` генерировала множество ошибок, связанных с правами доступа. С началом использования `slocate` эти ошибки — дело прошлого.

Особенности работы `slocate` видны на следующем примере:

```
$ locate slocate.db
$ su -
# locate slocate.db
/var/lib/slocate/slocate.db.tmp
/var/lib/slocate/slocate.db
```

Поскольку сначала вы были зарегистрированы как обычный пользователь (не `root`), первая попытка поиска окончилась неудачей. Используя команду `su`, вы переключились для работы от имени `root`, а затем снова запусти-

ли locate. Теперь поиск дал результаты (slocate.db — это база данных, используемая slocate).

Для удобства пользователей во многих системах создают ссылку /usr/bin/locate, которая реально указывает на /usr/bin/slocate. Для того чтобы выяснить, сделано ли это в вашей системе, попробуйте выполнить следующую команду (в данном примере приведены результаты, полученные в K/Ubuntu 5.10):

```
$ ls -l /usr/bin/locate
root  root  /usr/bin/locate -> slocate
```

Поскольку пользователь может и не знать, что вызывает именно slocate, а также потому, что в имени locate на одну букву меньше и ее быстрее вводить, мы будем в данной книге говорить о locate, не задумываясь о том, какая программа выполняется реально.

Поиск в базе имен файлов без учета регистра

```
locate -i
```

В предыдущем примере мы искали файлы или каталоги, содержащие в своем имени последовательность символов haggard. Полученные результаты выглядели следующим образом:

```
$ locate haggard
.../txt/rider_haggard
.../txt/rider_haggard/Queen_of_the_Dawn.txt
.../txt/rider_haggard/Allan_and_the_Ice-Gods.txt
.../txt/rider_haggard/Heu-Heu_or_The_Monster.txt
```

Они были получены потому, что именно этот фрагмент текста составлял часть имени каталога. Но если бы каталог назывался H_Rider_Haggard, поиск не дал бы результатов. Когда мы задаем опцию -i, при поиске не учитывается

регистр символов. В этом случае были бы найдены файлы, в названии пути к которым содержалось бы haggard или Haggard (а также HAGGARD и даже HaGgArD).

```
$ locate -i haggard
/txt/rider_haggard
/txt/rider_haggard/Queen_of_the_Dawn.txt
/txt/rider_haggard/Allan_and_the_Ice-Gods.txt
/txt/rider_haggard/Heu-Heu_or_The_Monster.txt
/txt/Rider_Haggard
/txt/Rider_Haggard/King_Solomons_Mines.txt
/txt/Rider_Haggard/Allan_Quatermain.txt
```

Оказывается, в системе есть два каталога, удовлетворяющие новому критерию поиска. Если, вызывая locate, вы хотите получить максимальное количество результатов, не забывайте задавать опцию -i, в противном случае вы можете пропустить именно те файлы или каталоги, которые ищете.

Управление результатами поиска в базе имен файлов

```
-n
```

Если вы часто используете команду locate, то наверняка попадали в ситуации, подобные следующей:

```
$ locate pdf
/etc/cups/pdftops.conf
/etc/xpdf
/etc/xpdf/xpdfrc-latin2
```

В этом примере информация усечена. На моем компьютере такой поиск дал 2373 результата. Это слишком много! Чтобы просмотреть их, лучше использовать такую конструкцию:

```
$ locate pdf | less
```

В данном случае вы организуете конвейерную обработку, используя результаты поиска с помощью `locate` в качестве входных данных программы `less` (вопросы передачи данных для обработки команде `less` рассматривались в главе 5). Теперь 2373 строки результатов можно просмотреть постранично.

Если вам нужны лишь первые x результатов, используйте опцию `-n`, указав после нее число требуемых результатов.

```
$ locate -n 3 pdf
/etc/cups/pdftops.conf
/etc/xpdf
/etc/xpdf/xpdfrc-latin2
```

Теперь объем информации достаточно мал, и не исключено, что это все, что вам надо. Не позволяйте `locate` генерировать большой объем данных; управляйте ее выходом в соответствии со своими целями.

Обновление базы, используемой программой locate

```
updatedb
```

В первом разделе этой главы, где лишь начинался разговор о `locate`, было упомянуто, что причина столь быстрой работы команды в том, что поиск реально выполняется в базе данных, содержащей имена файлов и каталогов. При инсталляции программа `locate` автоматически настраивается на просмотр жесткого диска и обновление базы. Обычно такое обновление происходит ночью. Это очень удобно, но такой подход не позволяет найти файлы, которые лишь недавно были помещены в файловую систему.

Предположим, например, что вы инсталлировали Rootkit Hunter (программу, которая выявляет средства, используемые злоумышленниками), а потом хотите получить информацию об установленных файлах. Команда `locate` не поможет вам, поскольку она ничего не знает об этих файлах и не будет знать о них до тех пор, пока база данных не обновится. При необходимости вы можете в любое время вручную задать обновление базы, используемой `locate`. Для этого надо вызвать команду `updatedb`. Поскольку данная команда индексирует практически каждый файл и каталог на вашем компьютере, для вызова ее вам необходимы права `root` (в системах типа K/Ubuntu можно также задать `sudo`).

```
# apt-get install rkhunter
# exit
$ locate rkhunter
$ su -
# updatedb
# exit
$ locate rkhunter
/usr/local/rkhunter
/usr/local/rkhunter/bin
/usr/local/rkhunter/etc
```

В предыдущем примере мы сначала установили `rkhunter` (пакет Rootkit Hunter), а затем завершили сеанс пользователя `root`. После этого был выполнен поиск `rkhunter`, но он не дал результатов. Мы снова выступили под именем `root`, запустили `updatedb` для просмотра жесткого диска, в результате чего `locate` получила информацию об изменениях, а затем завершили сеанс `root`. И наконец, мы снова выполнили поиск `rkhunter` с помощью программы `locate` и на этот раз он был успешным.

Запуская updatedb, необходимо учитывать следующее: время работы этой программы прямо пропорционально числу файлов на вашем жестком диске и быстродействию компьютера. Если у вас быстрый процессор, быстрый диск и немного файлов, updatedb завершится быстро. А что если быстродействие процессора мало, скорость обмена с жестким диском невелика, а на диске миллионы файлов? Тогда придется запастись терпением. Если вы хотите узнать, сколько времени потребовалось для индексирования содержимого файловой системы, задайте перед updatedb команду time так, как показано ниже.

```
# time updatedb
```

Когда updatedb завершит работу, time сообщит о затраченном времени. Эти сведения будут полезны на случай, если вы не обладаете достаточным запасом времени и вам надо решить, имеет ли смысл вызывать updatedb.

На заметку

Команда updatedb дает те же результаты, что и запуск slocate с опцией -u. Более того, как нетрудно выяснить, updatedb — это всего лишь ссылка на slocate.

```
$ ls -l /usr/bin/updatedb
root root /usr/bin/updatedb -> slocate
```

Поиск фрагментов текстового файла

grep

Команда locate позволяет организовать поиск имен файлов и каталогов, но не дает возможности искать информацию в составе файлов. Для того чтобы решить эту задачу, надо использовать команду grep. При работе с этой командой ей задается шаблон поиска и файл или группа файлов

(или даже весь жесткий диск), в которых надо найти фрагмент, соответствующий шаблону. В результате выполнения grep отображает строки, в которых присутствует интересующий вас фрагмент.

```
$ grep pain three_no_more_forever.txt
all alone and in pain
```

В данном случае мы используем команду grep для проверки, содержится ли в указанном файле слово pain. Как видите, результат проверки положительный; grep выводит на экран строку, содержащую данное слово. А можно ли выполнить поиск в нескольких файлах? Сделать это позволяют символы групповых операций.

```
$ grep pain *
fiery inferno in space.txt:watch the paint peel,
three_no_more_forever.txt:all alone and in pain
the speed of morning.txt:of a Chinese painting.
8 hour a day.txt:nice paint job too
ghost pain.txt:Subject: ghost pain
```

Заметьте, что команда grep нашла все вхождения ключевого слова pain, в том числе paint и painting. Также обратите внимание на то, что данная команда выводит не только строки, содержащие искомое слово, но и имена файлов. До сих пор мы не испытывали трудностей, выполняя поиск посредством grep. Пора усложнить задачу.

Общие сведения о шаблонах поиска

Из предыдущего раздела вы узнали, что программа grep позволяет находить указанное слово в группе файлов. Это одно из самых простых применений grep. Теперь попробуем глубже разобраться в структуре шаблонов, используемых для поиска. При формировании этих шаблонов при-

меняется один из самых мощных инструментов Linux: регулярные выражения. Для того чтобы в полной мере воспользоваться возможностями, предоставляемыми программой grep, надо изучить механизм регулярных выражений. Однако, для того, чтобы подробно рассмотреть этот вопрос, потребовалась бы отдельная книга, поэтому здесь мы обсудим лишь общие положения.

Совет

Если вы хотите получить дополнительную информацию о регулярных выражениях, ее легко найти в глобальной сети. Но на мой взгляд, самым лучшим пособием будет книга Бена Форты *Освой самостоятельно регулярные выражения. 10 минут на урок* (ИД "Вильямс", 2004 г.).

Одна из особенностей, затрудняющих начинающим пользователям изучение grep, состоит в том, что существует несколько версий данной команды. Сведения о них приведены в табл. 9.1.

Таблица 9.1. Различные версии программы grep

Интерпретация шаблона	Опция команды grep	Отдельная команда
Основные регулярные выражения	`grep -G` (или `--basic-regexp`)	`grep`
Расширенные регулярные выражения	`grep -E` (или `--extended-regexp`)	`egrep`
Набор фиксированных строк, для каждой из которых может быть установлено соответствие	`grep -F` (или `--fixed-strings`)	`fgrep`
Регулярные выражения Perl	`grep -P` (или `--perl-regexp`)	Отсутствует

Как видно из таблицы, сама по себе команда `grep` поддерживает основные регулярные выражения. Если вы зададите опцию `-E` (или `--extended-regexp`) либо команду `egrep`, то сможете использовать расширенные регулярные выражения. Именно такие выражения вам потребуются в большинстве случаев, за исключением разве что очень простых критериев поиска. Более сложные варианты команды — это `grep` с опцией `-F` (или `--fixed-strings`) или команда `fgrep`, позволяющая задавать множественные условия поиска, и `grep` с опцией `-P` (или `--perl-regexp`), которая дает возможность применять конструкции, типичные для языка Perl.

На заметку

В данной книге мы будем в основном применять обычную команду `grep`, поддерживающую основные регулярные выражения. Отклонения от этого правила будут оговариваться отдельно.

Перед тем как продолжить изучение материала, надо обсудить некоторые особенности, которые могут затруднить его восприятие.

Символы групповых операций и регулярные выражения — не одно и то же. Например, и в том и в другом случае используется символ *, однако он интерпретируется поразному. Если в групповых операциях знаки ? и * означают произвольные символы, то в групповых операциях они управляют повторением предшествующей им конструкции. Например, символ групповой операции ? в `c?t` заменяется одним и только одним символом. Этому выражению соответствуют слова `cat`, `cot` и `cut`, но не `ct`. В регулярных выражениях символ ? в конструкции `c[a-z]?t` указывает на то, что буква из диапазона `a-z` может присутствовать один раз или отсутствовать вовсе. Такому шаблону соответствуют последовательности символов `cat`, `cot`, `cut`, а также `ct`.

Совет

Чтобы лучше понять различия между символами групповых операций и регулярными выражениями, обратитесь к документам *What is a Regular Expression* (`http://docs.kde.org/stable/en/kdeutils/KRegExpEditor/whatIsARegExp.html`), *Regular Expressions Explained* (`http://www.castaglia.org/proftpd/doc/contrib/regexp.html`) и *Wildcards Gone Wild* (`http://www.linux-mag.com/2003-12/power_01.html`).

Источником проблем при работе с командой `grep` могут также стать символы, имеющие специальное назначение. Например, выражение `[a-e]` обозначает диапазон символов, ему соответствует только одна из следующих букв: a, b, c, d или e. Используя `[` или `]` в регулярном выражении, надо различать случаи, когда они обозначают границы диапазона или непосредственно входят в последовательность символов, предназначенную для поиска. Символы, имеющие специальное значение, приведены ниже.

```
. ? [ ] ^ $ | \
```

И наконец, одинарная и двойная кавычки в регулярных выражениях существенно отличаются друг от друга. Одинарная кавычка сообщает о том, что должна быть найдена строка символов, а двойная кавычка указывает на использование переменных оболочки. Рассмотрим строку `hey you!`, заданную в качестве условия поиска.

```
$ grep hey you! *
grep: you!: No such file or directory
txt/pvzm/8 hours a day.txt:hey you! let's run!
txt/pvzm/friends & family.txt:in patience they wait
txt/pvzm/speed of morning.txt:they say the force
```

Поскольку в данном случае последовательность символов `hey you` была включена в состав команды без ограничи-

телей, команда grep неправильно обработала запрос. Она интерпретировала первое слово, hey, как ключевое, а второе слово, you!, как имя файла. Такого файла не существует, поэтому поиск в нем не может быть выполнен. Затем, восприняв символ *, она стала искать слово hey в каждом файле, содержащемся в текущем каталоге, и вернула три строки результатов. В первой из этих строк содержится фраза, которую требовалось найти, но сказать о том, что поиск был выполнен корректно, нельзя. Повторим попытку.

Ограничим последовательность символов для поиска двойными кавычками. Будет ли устранена проблема, возникшая в предыдущем случае?

```
$ grep "hey you!" *
bash: !" *: event not found
```

Стало еще хуже! Включив в выражение двойные кавычки, мы допустили большую ошибку и в результате не получили даже тех строк, которые команда grep нашла в прошлый раз. Что случилось? Символ ! — это команда оболочки, ссылающаяся на список предыстории. По правилам за ним должен следователь идентификатор процесса, представляющий команду, выполненную ранее, например !264. Таким образом, bash ожидает идентификационный номер после символа ! и сообщает, что не может найти команду " * (двойная кавычка, пробел и звездочка).

Дело в том, что двойная кавычка указывает на использование в шаблоне поиска переменных оболочки, хотя мы совсем не собирались применять их. Таким образом, двойные кавычки в данном случае неуместны. Попробуем использовать одинарные кавычки.

```
$ grep 'hey!' *
txt/pvzm/8 hours a day.txt:hey you! let's run!
```

Теперь все намного лучше. Одинарные кавычки сообщают grep не о переменных оболочки, а лишь о том, что критерием поиска является обычная строка символов. Результатом является строка, содержащая указанное выражение, чего мы и добивались.

Какие можно сделать выводы? Необходимо знать, когда использовать одинарные кавычки, когда двойные, а когда можно обойтись без них. Если вы ищете конкретную последовательность символов, ее надо поместить в одинарные кавычки. Если же вы хотите включить в эту последовательность переменную оболочки (такая необходимость возникает крайне редко), следует использовать двойные кавычки. Если же вы ищете одно слово, содержащее лишь числа и буквы, вполне можно обойтись без кавычек. Желая перестраховаться, поместите слово в одинарные кавычки — вреда от этого не будет.

Рекурсивный поиск фрагментов текста в файлах

```
-R
```

Символ групповой операции * позволяет указать несколько файлов в одном каталоге, но для поиска в нескольких подкаталогах вам понадобится опция -R (или --recursive). Попробуем найти слово hideous, которое любили употреблять авторы романов ужасов в XIX и начале XX столетия. Поиск будем производить в наборе файлов, содержащем литературные произведения.

```
$ grep -R hideous *
machen/great_god_pan.txt:know, not in
your most fantastic, hideous dreams can you have
machen/great_god_pan.txt:hideously
contorted in the entire course of my practice, and
```

```
machen/great_god_pan.txt:death was
horrible. The blackened face, the hideous form upon
lovecraft/Beyond the Wall of Sleep.txt:some
hideous but unnamed wrong, which
lovecraft/Beyond the Wall of Sleep.txt:blanket
over the hideous face, and awakened the nurse.
lovecraft/Call of Cthulhu.txt:hideous a chain. I
think that the professor, too, intended to
lovecraft/Call of Cthulhu.txt:voodoo meeting;
and so singular and hideous were the rites
lovecraft/Call of Cthulhu.txt:stated, a very
crude bas-relief of stone, comprising a hideous...
```

Совет

Если вы получите слишком много результатов, вы можете передать их средствами конвейерной обработки программе `less`.

```
$ grep -R hideous * | less
```

Можно также перенаправить вывод команды в текстовый файл, а потом просмотреть его содержимое с помощью текстового редактора.

```
$ grep -R hideous * > hideous_in_horror.txt
```

Этот способ хорош тем, что вы сможете в любой момент вернуться к полученным результатам, не выполняя снова команду `grep`.

Поиск фрагментов текста в файлах без учета регистра

```
-i
```

По умолчанию при поиске, осуществляемом посредством команды `grep`, учитывается регистр символов. В предыдущем разделе мы искали слово `hideous`. А как насчет `Hideous`?

```
$ grep Hideous h_p_lovecraft/*
h_p_lovecraft/the_whisperer_in_darkness.txt: them.
Hideous though the idea was. I knew...
```

Поиск по слову `hideous` дал 463 результата, а по слову `Hideous` — один. Можно ли задать поиск обоих слов? Это позволяет опция `-i` (или `--ignore-case`), которая указывает, что при поиске не должен учитываться регистр, поэтому будут найдены не только `hideous` и `Hideous`, но также `HiDeOuS` и `HIDEOUS` и другие сочетания строчных и прописных букв, если они, конечно, будут встречаться в тексте.

```
$ grep -i hideous h_p_lovecraft/*
h_p_lovecraft/Call of Cthulhu.txt:voodoo meeting;
and so singular and hideous were the rites
h_p_lovecraft/Call of Cthulhu.txt:stated, a very
crude bas-relief of stone, comprising a hideous
h_p_lovecraft/the_whisperer_in_darkness.txt: them.
Hideous though the idea was, I knew...
```

При этом надо учитывать, что в общем случае при использовании опции `-i` число результатов может увеличиться в несколько раз и даже на порядки. В конце предыдущего раздела были приведены рекомендации, как справиться со слишком большим объемом информации, возвращаемой командой `grep`.

Поиск слов в файлах

```
-w
```

Вспомним раздел, в котором мы только начали изучать команду `grep`. Мы искали слово `pain`, а команда `grep` вернула нам список строк, в которых встречалась данная последовательность символов.

```
$ grep pain *
fiery inferno in space.txt:watch the paint peel,
three_no_more_forever.txt:all alone and in pain
the speed of morning.txt:of a Chinese painting.
8 hour a day.txt:nice paint job too
ghost pain.txt:Subject: ghost pain
```

По умолчанию grep находит все вхождения указанной строки, в данном случае pain, в том числе слова paint и painting. Если бы в тексте присутствовали слова painless, Spain или painstaking, они также были бы учтены. Но что делать, если необходимы только те строки, в которые входит именно слово pain? Для этой цели предназначена опция -w (или --word-regexp).

```
$ grep -w pain *
three_no_more_forever.txt:all alone and in pain
ghost pain.txt:Subject: ghost pain
```

Данная опция позволяет сузить поиск и соответственно уменьшить набор результатов.

Отображение номеров строк

```
-n
```

Команда grep отображает каждую строку, содержащую заданную последовательность символов, но не сообщает о том, в какой части файла расположена эта строка. Чтобы получить номер строки, надо использовать опцию -n (или --line-number).

```
$ grep -n pain *
fiery inferno in space.txt:56:watch the paint peel,
three_no_more_forever.txt:19:all alone and in pain
the speed of morning.txt:66:of a Chinese painting.
8 hour a day.txt:78:nice paint job too
ghost pain.txt:32:Subject: ghost pain
```

Теперь, когда мы знаем номера строк, содержащих последовательность символов `pain`, достаточно просто, используя любой текстовый редактор, обратиться непосредственно к нужной строке.

Поиск слов в выходных данных других команд

```
$ ls -1 | grep 1960
```

Команда `grep` сама по себе является мощным средством поиска, но ее также можно использовать как фильтр для обработки выходных данных, сгенерированных другими программами. Предположим, например, что вы храните mp3-файлы с записями Джона Колтрейна, выделив для каждого альбома отдельный подкаталог, причем имя подкаталога начинается с года, в котором соответствующий альбом был выпущен. Часть информации, полученной с помощью команды `ls -1`, приведена ниже (опция `-1` приводит к тому, что в каждой строке отображается только одно имя).

```
$ ls -1
1956_Coltrane_For_Lovers
1957_Blue_Train
1957_Coltrane_[Prestige]
1957_Lush_Life
1957_Thelonious_Monk_With_John_Coltrane
```

Предположим теперь, что вам необходимо получить список альбомов, выпущенных в 1960 году. Решить эту задачу можно, передав результаты выполнения `ls -1` команде `grep`.

```
$ ls -1 | grep 1960
1960_Coltrane_Plays_The_Blues
1960_Coltrane's_Sound
```

```
1960_Giant_Steps
1960_My_Favorite_Things
```

Немного поразмыслив, вы придумаете сотни вариантов применения команды grep. Рассмотрим еще один пример. Команда ps выводит список выполняющихся процессов, причем опция -f указывает ps на то, что необходимо отобразить полный список с подробной информацией о каждом процессе, а опция -U, за которой следует пользовательское имя, ограничивает вывод лишь процессами, принадлежащими конкретному пользователю. Задав -fU scott, мы, вероятнее всего, получим длинный список, в котором очень сложно будет найти информацию о требуемом процессе. Команда grep позволяет сократить объем выходных данных.

На заметку

Для экономии места часть информации, обычно отображаемой по команде ps, здесь не приводится.

```
$ ps -fU scott | grep firefox
scott 17623 /bin/sh /opt/firefox/firefox
scott 17634 /opt/firefox/firefox-bin
scott  1601 grep firefox
```

Команда ps выводит информацию о всех процессах, принадлежащих пользователю scott (в данном случае их 64), но выходные данные посредством механизма конвейерной обработки передаются программе grep, отсеивающей все строки, в которых не содержится слово firefox. К сожалению, последняя строка лишняя: нам нужна информация о самой программе firefox, а не о ее имени, переданном команде grep. Скрыть эту строку можно следующим образом:

```
$ ps -fU scott | grep [f]irefox
scott 17623 /bin/sh /opt/firefox/firefox
scott 17634 /opt/firefox/firefox-bin
```

Теперь первая буква слова, передаваемого `grep`, лежит в диапазоне от `f` до `f`. Этому критерию соответствуют строки, сгенерированные программой `ps`, которые содержат слово `firefox`. Последовательность `[f]irefox`, переданная `grep`, не удовлетворяет условиям поиска. поскольку в ней содержатся символы `[` и `]`. Сама же команда `grep` использует шаблон `[f]irefox`, которому соответствует только слово `firefox`. Такой способ выглядит несколько сложно, но он дает нужные результаты. Возьмите его на вооружение!

Просмотр контекста для слов, имеющихся в файлах

```
-A, -B, -C
```

Когда речь идет о работе с данными, важность контекста трудно переоценить. Как вы знаете, программа `grep` выводит строку, содержащую заданную последовательность символов, но мы также можем указать `grep` выводить предыдущие и последующие строки. В предыдущем разделе `grep` использовалась для обработки списка альбомов Джона Колтрейна. Одним из наилучших считается "A Love Supreme". Какие три альбома были выпущены перед ними? Чтобы получить ответ, зададим опцию `-B` (или `--before-context=#`).

```
$ ls -1 | grep -B 3 A_Love_Supreme
1963_Impressions
1963_John_Coltrane_&_Johnny_Hartman
1963_Live_At_Birdland
1964_A_Love_Supreme
```

Если вас интересует, какие альбомы последовали за A Love Supreme, задайте опцию -A (или --after-context=#).

```
$ ls -1 | grep -A 3 'A_Love_Supreme'
1964_A_Love_Supreme
1964_Coltrane's_Sound
1964_Crescent
1965_Ascension
```

Для того чтобы получить полный контекст, можно использовать опцию -C (или --context=#), которая является сочетанием двух опций, рассмотренных выше.

```
$ ls -1 | grep -C 2 'A_Love_Supreme'
1963_John_Coltrane_&_Johnny_Hartman
1963_Live_At_Birdland
1964_A_Love_Supreme
1964_Coltrane's_Sound
1964_Crescent
```

Такая информация в некоторых случаях может быть сложной для восприятия, поскольку при обнаружении соответствия условиям поиска программа отображает сразу несколько строк. Так, например, в названиях ряда альбомов Колтрейна присутствует слово "live", и если вы зададите вывод предшествующих и последующих альбомов, результаты могут оказаться несколько неожиданными.

```
$ ls -1 | grep -C 1 Live
1963_John_Coltrane_&_Johnny_Hartman
1963_Live_At_Birdland
1964_A_Love_Supreme
--
1965_Last_Trane
1965_Live_in_Seattle
1965_Major_Works_of_John_Coltrane
--
```

```
1965_Transition
1966_Live_at_the_Village_Vanguard_Again!
1966_Live_in_Japan
1967_Expression
1967_Olatunji_Concert_Last_Live_Recording
1967_Stellar_Regions
```

Символы -- отделяют одну группу результатов от другой. Первые две группы очевидны — рядом с альбомом со словом "Live" в названии находятся другие альбомы, однако последняя группа выглядит гораздо сложнее. Несколько альбомов со словом "Live" непосредственно следуют друг за другом, поэтому результаты объединены. Это может показаться странными, но если вы ищете каждое вхождение слова "Live", то заметите, что отображается информация о предшествующем и последующем альбомах.

Результаты станут еще более информативными, если мы дополнительно укажем при вызове команды опцию -n, которая задает вывод номеров строк.

```
$ ls -1 | grep -n -C 1 Live
37-1963_John_Coltrane_&_Johnny_Hartman
38:1963_Live_At_Birdland
39-1964_A_Love_Supreme
--
48-1965_Last_Trane
49:1965_Live_in_Seattle
50-1965_Major_Works_of_John_Coltrane
--
52-1965_Transition
53:1966_Live_at_the_Village_Vanguard_Again!
54:1966_Live_in_Japan
55-1967_Expression
56:1967_Olatunji_Concert_Last_Live_Recording
57-1967_Stellar_Regions
```

Теперь опция -C предоставляет еще больше информации о каждой строке: отображает после номера строки разные символы. Символ : указывает на то, что строка соответствует условиям поиска, а символ - означает, что это предшествующая или последующая строка. Строка 54:1966_Live_in_Japan выполняет две функции: следует за строкой 53:1966_Live_at_the_Village_Vanguard_Again!, по этому критерию после номера должен быть указан символ -, и сама содержит последовательность символов, заданную при вызове команды, т.е. в ней должно быть двоеточие. Поскольку соответствие критерию поиска является более важным признаком, предпочтение отдается символу :.

Отображение строк, не содержащих указанных слов

```
-v
```

С момента смерти Джона Колтрейна прошло около 40 лет, но его альбомы по-прежнему популярны. Знатоки искусства также высоко ценят творчество группы "Led Zeppelin", которая выпустила девять альбомов; в названиях многих из них присутствует имя группы (вы скажете, что четвертый альбом вовсе не имеет названия, но большинство критиков называют его "Led Zeppelin IV"). Предположим, что вы хотите получить список каталогов, в которые поместили альбомы "Led Zeppelin", но в названии которых не содержатся сами слова "Led Zeppelin". Опция -v (или --invert-match) позволяет вам отобразить только те результаты, которые не соответствуют заданному шаблону.

```
$ ls -1
1969_Led_Zeppelin
1969_Led_Zeppelin_II
```

```
1970_Led_Zeppelin_III
1971_Led_Zeppelin_IV
1973_Houses_Of_The_Holy
1975_Physical_Graffiti
1976_Presence
1979_In_Through_The_Out_Door
1982_Coda
$ ls -1 | grep -v Led_Zeppelin
1973_Houses_Of_The_Holy
1975_Physical_Graffiti
1976_Presence
1979_In_Through_The_Out_Door
1982_Coda
```

Опция `-v` дает возможность сократить набор результатов и отобразить только те из них, которые вам действительно нужны. Вряд ли вам придется использовать данную опцию очень часто, но в некоторых случаях она может оказаться полезной.

Отображение списка файлов, содержащих указанное слово

```
-l
```

Как уже неоднократно говорилось, команда `grep` выводит строки, содержащие слово, заданное для поиска. Однако в некоторых случаях строки бывают излишни; нужны лишь имена файлов, в которых находится слово. В одном из предыдущих разделов мы искали вхождение слова `hideous`. Опция `-l` (или `--files-with-matches`) дает возможность получить список файлов (опция `-i`, указанная в той же строке, задает поиск без учета регистра символов).

```
$ grep -il hideous h_p_lovecraft/*
h_p_lovecraft/Call of Cthulhu.txt
h_p_lovecraft/From Beyond.txt
h_p_lovecraft/The Case of Charles Dexter Ward.txt
```

Подобные результаты в особенности полезны в сочетании с другими командами. Например, если вы хотите вывести на печать список файлов из конкретного каталога, в которых встречается слово `hideous`, можете объединить команды `grep` и `lpr` следующим образом:

```
$ grep -il hideous h_p_lovecraft/* | lpr
```

Помните, что данная команда выведет список имен файлов, а не их содержимое (если вы хотите распечатать сами тексты, вам придется использовать команду `cat`).

Поиск слов в результатах поиска

```
grep | grep
```

Предположим, что нам надо получить список альбомов Джона Колтрейна, выпущенных в последние три года его творческой биографии. Сделать это достаточно просто.

```
$ ls -1 | grep 196[6-7]
1966_Live_at_the_Village_Vanguard_Again!
1966_Live_in_Japan
1967_Expression
1967_Olatunji_Concert_Last_Live_Recording
1967_Stellar_Regions
```

Диапазон значений `[5-7]` ограничивает список годами 1965–1967; в противном случае его размеры были бы очень велики. Пока все хорошо, но допустим, что нам надо, чтобы в этот список не входили альбомы, в названии которых есть слово `Live` (мы вряд ли поступаем правильно, отка-

зываясь от данных произведений, но ведь это всего лишь пример)?

Ниже представлено выражение, которое решает эту задачу.

```
$ ls -1 | grep 196[6-7] | grep -v Live
1967_Expression
1967_Stellar_Regions
```

Опция `-v`, рассмотренная нами ранее в этой главе, отсеивает результаты, содержащие слово "Live", но в данном случае нас интересует не это. Главное для нас — тот факт, что мы получаем выходные данные команды `ls -1`, передаем их команде `grep 196[5-7]`, а затем новые результаты снова подвергаем фильтрации посредством второй команды `grep`, на этот раз вызванной с опцией `-v Live`. В конце концов мы получаем именно то, что нам надо: список альбомов Джона Колтрейна, выпущенных в 1965–1967 годах, в названии которых не содержится слово "Live". Вот вы и ознакомились с еще одним набором возможностей, доступным из командной строки Linux.

Выводы

В данной главе внимание было уделено двум командам, которые часто используются на практике: `locate` и `grep`. Несмотря на то что эти команды принадлежат одной группе, реализующей средства поиска информации на компьютере, они решают разные задачи. Команда `locate` выполняет поиск по именам файлов, используя для ускорения работы специальную базу данных, а команда `grep` просматривает в реальном времени содержимое файлов и извлекает фрагменты, удовлетворяющие условиям поиска.

И команда `locate`, и команда `grep` — удобные средства, но возможности поиска в файловой системе далеко не

исчерпываются ими. Следующая глава посвящена одной из самых мощных и гибких команд системы Linux, которая дополняет `locate` и `grep`, а также может работать совместно с ними. Это команда `find`. Переходите к следующей главе, и начнем знакомство с ней.

10

Команда find

В предыдущей главе мы рассматривали команды, позволяющие выполнять поиск файлов (`locate`) и данных в составе файлов (`grep`). Третья команда из той же группы — `find`. Если `locate` выполняет поиск в базе данных имен файлов, что позволяет ей работать быстро, но ставит в зависимость от времени обновления базы, то команда `find` непосредственно ищет файлы по заданным условиям. Поскольку команда `find` осуществляет реальный обход дерева каталогов, она работает намного медленнее, чем `locate`, однако предоставляет возможности, недоступные посредством команды `locate`.

В этой главе мы будем выполнять поиск на внешнем жестком диске, смонтированном в каталоге `/media/music`. Вы увидите, что команда `find` позволяет обрабатывать самые разнообразные запросы.

Поиск файлов по имени

```
find -name
```

Чаще всего команда `find` используется для поиска файлов по имени или его части. Для этой цели предусмотрена опция `-name`. По умолчанию поиск осуществляется рекурсивно в дереве каталогов. Давайте найдем mp3-файлы с записями группы `Shaggs`.

```
$ cd /media/music
$ find . -name Shaggs
./Outsider/Shaggs
```

Что-то не так! Команда `find` нашла каталог, а не файлы, содержащиеся в нем. Почему это произошло? Дело в том, что мы не указали символы групповых операций, поэтому команда `find` искала файлы по точному совпадению имени со строкой `Shaggs`. Такое имя имеет только один файл — каталог, содержащий mp3-файлы (как вы, наверное, помните, каталог — это файл специального типа).

Нам надо использовать символ групповой операции, но чтобы предотвратить специальную интерпретацию его оболочкой, поместим имя, содержащее этот символ, в кавычки. Попробуем выполнить поиск снова.

```
$ find . -name "*Shaggs*"
./Outsider/Shaggs
./Outsider/Shaggs/Gimme_Dat_Ting_(Live).mp3
./Outsider/Shaggs/My_Pal_Foot_Foot.ogg
./Outsider/Shaggs/I_Love.mp3
./Outsider/Shaggs/Things_I_Wonder.ogg
```

Теперь мы получили информацию не только о каталоге, но и о файлах.

На заметку

Не подозревая о том, мы использовали еще одну опцию: `-print`. Эта опция сообщает программе `find`, что результаты поиска надо вывести на терминал. По умолчанию эта опция включена, и ее не надо указывать явно при вызове `find`.

Важно отметить еще одну особенность программы `find`: формат представления результатов зависит от пути, указанного при вызове команды. В предыдущем примере мы использовали относительный путь — указали текущий ка-

талог, поэтому результаты также были представлены относительно текущего каталога. Что произойдет, если мы зададим абсолютный путь, начинающийся с символа /?

```
$ find / -name "*Shaggs*"
/music/Outsider/Shaggs
/music/Outsider/Shaggs/Gimme_Dat_Ting_(Live).mp3
/music/Outsider/Shaggs/My_Pal_Foot_Foot.ogg
/music/Outsider/Shaggs/I_Love.mp3
/music/Outsider/Shaggs/Things_I_Wonder.ogg
```

При поиске по абсолютному пути результаты также представляются посредством абсолютного пути. Другие применения этого принципа мы увидим далее в данной главе. Сейчас же просто запомним эту особенность.

Поиск файлов по имени владельца

```
find - user
```

Помимо поиска по имени файла, можно также осуществлять поиск по имени владельца. Предположим, что вы хотите найти файлы, принадлежащие пользователю `scott`. Задайте при вызове команды `find` опцию `-user`, введите после нее имя пользователя (или числовой идентификатор, который можно найти в файле `/etc/passwd`).

```
$ find . -user scott
```

Число результатов слишком велико! Возможно, будет проще искать файлы, которые не принадлежат пользователю `scott`. Сделать это можно, указав символ ! перед опцией, действие которой вы хотите изменить на обратное.

На заметку

Для экономии места некоторые данные, отображаемые по команде `ls -l`, здесь не приводятся.

```
$ find . ! -user scott
./Outsider/Wing/01_-_Dancing_Queen.mp3
$ ls -l ./Outsider/Wing/01_-_Dancing_Queen.mp3
gus-music ./Outsider/Wing/01_-_Dancing_Queen.mp3
```

Один из файлов с записью композиций в исполнении Винг принадлежит не пользователю scott, а gus. Для того чтобы исправить положение, используем команду chown, которая была рассмотрена в главе 7. Помните, что при необходимости вы всегда можете использовать символ ! в качестве оператора NOT. Так, только что мы указали команде find найти файлы, не принадлежащие пользователю scott.

Поиск файлов по имени группы

```
file -group
```

Если в команде предусмотрена опция для работы с пользователями, логично предположить, что поддерживается также и опция для работы с группами. И действительно, в команде find такая опция есть. Если вы хотите найти файлы, принадлежащие определенной группе, вам надо указать опцию -group и ввести после нее имя или номер группы. На диске, используемом при поиске, владельцем файлов является пользователь scott, а в качестве группы указана music. Попробуем найти файлы, не принадлежащие группе music.

```
$ find . ! -group music
./Disco/Brides_of_Funkenstein-Disco_to_Go.mp3
./Disco/Sister_Sledge-He's_The_Greatest_Dancer.mp3
./Disco/Wild_Cherry-Play_That_Funky_Music.mp3
./Electronica/New_Order/Bizarre_Love_Triangle.mp3
```

Из огромного числа файлов только четыре не принадлежат данной группе. Вызовем команду `chgrp`, которая обсуждалась в главе 7, и изменим их принадлежность группе. Теперь информация на диске приведена в порядок.

Заметьте, что мы снова использовали символ `!`, чтобы выполнить поиск файлов, не принадлежащих группе `music`.

Поиск файлов по размеру

```
file -size
```

В некоторых случаях приходится использовать в качестве критерия для поиска файлов их размер. Команда `find` позволяет решить и эту задачу. Для того чтобы указать размер, надо задать опцию `-size` и ввести после нее букву, которая представляет используемую вами схему размера. Если вы не зададите букву, будет использовано значение по умолчанию, но следует иметь в виду, что оно не обязательно будет соответствовать вашим намерениям. По умолчанию, если после числа вы не введете букву, принимается значение размера в байтах, деленное на 512 и округленное вверх до ближайшего целого числа. Некоторым пользователям покажется, что здесь слишком много математики и проще указать суффикс, представляющий единицу изменения размера. Набор допустимых суффиксов представлен в табл. 10.1.

Таблица 10.1. Суффиксы, используемые при поиске файлов по размеру

Суффикс	Значение
`b`	512-байтовые блоки (по умолчанию)
`c`	Байты
`k`	Килобайты
`M`	Мегабайты
`G`	Гигабайты

Предположим, что нам надо найти файлы с записью песен группы "Clash" из знаменитого альбома "London Calling", причем размер файлов должен составлять 10 Мбайт (конечно же, использовалось кодирование, обеспечивающее наивысшее качество). Эта задача легко решается с помощью команды find.

```
$ cd Punk/Clash/1979_London_Calling
$ find . -size 10M
./07_-_The_Right_Profile.ogg
./08_-_Lost_In_The_Supermarket.ogg
./09_-_Clampdown.ogg
./12_-_Death_Or_Glory.ogg
```

Результаты выглядят странно. Только четыре файла? Чтобы понять причины такого поведения команды find, надо учесть следующую ее особенность: если вы задаете значение 10 Мбайт, команда find ищет файлы, размер которых в точности равен 10 Мбайт (конечно же, с учетом округления). Если вам нужны файлы размером больше 10 Мбайт, то надо перед значением размера ввести символ +, если же размеры файлов должны быть меньше 10 Мбайт, следует задать знак -.

```
$ find . -size +10M
./Jimmy_Jazz.ogg
./Lover's_Rock.ogg
./Revolution_Rock.ogg
```

И снова возникла проблема. Когда мы указали размер для поиска, равный 10 Мбайт, были найдены файлы, размер которых равен этому значению, но не больше его, а когда задали +10M, получили список файлов, размер которых превышает 10 Мбайт, но файлы, размер которых в точности равен 10 Мбайт, не были учтены. А как получить сведения об обоих наборах файлов? Ответ на этот вопрос вы найдете ниже в данной главе.

Совет

Если вы хотите отыскать большие текстовые файлы, используйте после числового значения букву с. Как показано в табл. 10.1, этот символ задает измерение размера в байтах. Каждый символ в текстовом файле занимает один байт, поэтому ясно, что символ с — это первая буква в слове characters.

Например, чтобы найти очень большой текстовый файл, можно использовать выражение

```
$ find /home/scott/documents -size +500000c
```

Поиск файлов по типу

```
find -type
```

Одна из самых полезных опций программы `find` — это `-type`, позволяющая указать тип объекта для поиска. В главе 1 было сказано, что все объекты в системе Unix являются файлами, поэтому если вы укажете программе `find` тип файла, она найдет для вас нужный объект. В табл. 10.2 перечислены типы файлов, поддерживаемые `find`.

Таблица 10.2. Обозначения при поиске файлов по типу

Буква, представляющая тип файла	Значение
f	Обычный файл
d	Каталог
l	Символьная ссылка
b	Специальный файл блочного типа
c	Специальный файл символьного типа
p	FIFO
s	Сокет

Предположим, нам надо получить список альбомов группы "Steely Dan". Поскольку каждый альбом помещен в отдельный каталог, ответ на интересующий нас вопрос даст программа find с опцией -d.

```
$ cd /media/music/Rock
$ find Steely_Dan/ -type d
Steely_Dan/
Steely_Dan/1980_Gaucho
Steely_Dan/1975_Katy_Lied
Steely_Dan/1974_Pretzel_Logic
Steely_Dan/1976_The_Royal_Scam
Steely_Dan/2000_Two_Against_Nature
Steely_Dan/1972_Can't_Buy_A_Thrill
Steely_Dan/1973_Countdown_To_Ecstasy
Steely_Dan/1977_Aja
```

Чтобы со списком было удобнее работать, отсортируем альбомы по годам. Это нетрудно сделать, так как имя каталога начинается с года.

```
$ cd /media/music/Rock
$ find Steely_Dan/ -type d | sort
Steely_Dan/
Steely_Dan/1972_Can't_Buy_A_Thrill
Steely_Dan/1973_Countdown_To_Ecstasy
Steely_Dan/1974_Pretzel_Logic
Steely_Dan/1975_Katy_Lied
Steely_Dan/1976_The_Royal_Scam
Steely_Dan/1977_Aja
Steely_Dan/1980_Gaucho
Steely_Dan/2000_Two_Against_Nature
```

Как видите, для фильтрации результатов, сгенерированных командой find, используется фильтр, информация которому передается посредством механизма конвейерной

обработки. Итак, вы уже можете много сделать с помощью команды find. По мере дальнейшего изучения материала ваши возможности еще больше увеличатся.

Отображение результатов при выполнении всех выражений (AND)

```
find -a
```

Возможность объединить несколько опций позволит еще более конкретизировать поиск. Опция -a (или -and) позволяет связать вместе столько опций, сколько вам необходимо. Предположим, например, что вы хотите получить список всех композиций группы "Rolling Stones". Вы можете использовать опцию -name "Rolling_Stones*", но она не даст необходимых результатов; некоторые имена, содержащие последовательность символов Rolling_Stones, принадлежат каталогам, а другие представляют сбой ссылки. Таким образом, нам надо также задать опцию -type f. Объединим вместе две опции.

```
$ find . -name "Rolling_Stones*" -a -type f
1968_Beggars_Banquet/03-Dear_Doctor.mp3
1968_Beggars_Banquet/01-Sympathy_For_The_Devil.mp3
1968_Beggars_Banquet/02-No_Expectations.mp3
1968_Beggars_Banquet/04-Parachute_woman.mp3
1968_Beggars_Banquet/05-Jig-Saw_Puzzle.mp3
1968_Beggars_Banquet/06-Street_Fighting_Man.mp3
```

Вроде бы все в порядке, но сколько композиций мы получили реально? Передадим результаты работы команды find программе wc (ее имя является сокращением от word count, но если задать опцию -l, она будет подсчитывать не слова, а строки).

```
$ find . -name "Rolling_Stones*" -a -type f | wc -l
317
```

Как видите, в нашем распоряжении 317 композиций "Rolling Stones". И выяснить это нам опять же помогли команды Linux.

Отображение результатов при выполнении любого из выражений (OR)

```
find -o
```

Ранее в данной главе мы использовали команду find для поиска всех записей Clash из альбома "London Calling", которые содержались в файлах размеров 10 Мбайт. Там же выполнялся поиск файлов, размер которых превышает 10 Мбайт, но объединить результаты опция -size не позволяла. Из предыдущего раздела вы узнали, что опция -a дает возможность объединять другие опции в виде логического выражения AND. Сейчас мы используем опцию -o (или -or) для реализации операции OR.

Итак, чтобы найти записи из альбома "London Calling", которые содержатся в файле размером не менее 10 Мбайт, мы используем следующую команду:

```
$ cd London_Calling
$ find . -size +10M -o -size 10M
03_-_Jimmy_Jazz.ogg
07_-_The_Right_Profile.ogg
08_-_Lost_In_The_Supermarket.ogg
09_-_Clampdown.ogg
12_-_Death_Or_Glory.ogg
15_-_Lover's_Rock.ogg
18_-_Revolution_Rock.ogg
(25th_Anniversary)-18-Revolution_Rock.mp3
(25th_Anniversary)-37-Heart_And_Mind.mp3
(25th_Anniversary)-39-London_Calling_(Demo).mp3
```

Снова проблема. Мы получили также сведения о 25-й годовщине "London Calling", а это не входило в наши планы. Нам надо сделать следующее: во-первых, исключить данные о 25-й годовщине, а во-вторых, убедиться, что выражение OR работает корректно.

Чтобы убрать записи, посвященные 25-й годовщине, мы включим в состав команды конструкцию ! -name "*25*". Чтобы обеспечить работу выражения OR, надо поместить его в круглые скобки. При этом перед скобками следует задать символы обратной косой черты, чтобы оболочка правильно интерпретирована их, кроме того, перед выражением и после него надо включить пробелы. В окончательном виде команда выглядит следующим образом:

```
$ cd London_Calling
$ find . \( -size +10M -o -size 10M \) !
  -name "*25*"
03_-_Jimmy_Jazz.ogg
07_-_The_Right_Profile.ogg
08_-_Lost_In_The_Supermarket.ogg
09_-_Clampdown.ogg
12_-_Death_Or_Glory.ogg
15_-_Lover's_Rock.ogg
18_-_Revolution_Rock.ogg
```

Наконец-то мы получили требуемые результаты — список из семи файлов размером не менее 10 Мбайт.

Мы также можем использовать опцию -o, чтобы выяснить, записи скольких песен хранятся на диске. Начнем поиск с корневого каталога для данного диска (/media/music) и используем опцию -a (немного терпения, и мы снова воспользуемся опцией -o).

```
$ find . -name "*mp3*" -a -type f | wc -l
23407
```

Двадцать три тысячи? Это неверно. Причина ясна. Мы искали только файлы mp3, но большое число записей представлено в формате Ogg Vorbis. Этот формат был разработан как проект с открытым кодом — альтернатива mp3. Выполним поиск файлов mp3 или ogg и подсчитаем результаты с помощью `wc -l`.

```
$ find . \( -name "*mp3*" -o -name "*.ogg*" \) -a
-type f | wc -l
41631
```

Это выглядит значительно лучше, однако на диске есть также файлы FLAC. Добавим еще одну опцию `-o`.

```
$ find . \( -name "*mp3*" -o -name "*.ogg*" -o -name
"*.flac*" \) -a -type f | wc -l
42187
```

Теперь мы знаем, что число файлов на диске превышает 42000. Скоро появятся новые!

Отображение результатов, если выражение не выполняется (NOT)

```
find -n
```

Мы уже использовали ранее символ ! для того, чтобы инвертировать значение выражения. Вернемся к рассмотрению этого оператора. В предыдущем разделе с помощью команды `find` мы определили, сколько файлов mp3, ogg и flac находятся на диске. А сколько всего файлов на диске?

```
$ find . -name "*" | wc -l
52111
```

Из 52000 файлов только 42000 mp3, ogg или flac. А зачем остальные? Сформируем команду `find`, которая исключит из поиска файлы, имена которых заканчиваются

на .mp3, .ogg или .flac. Также нам не следует учитывать каталоги. Поместим эти четыре условия в круглые скобки и зададим перед всем выражением символ !. Таким образом мы отобразим лишь объекты, не удовлетворяющие четырем описанным выше условиям.

```
$ find . ! \( -name "*mp*" -o -name "*ogg" -o
  -name "*flac" -o -type d \)
./Folk/Joan_Baez/Joan_Baez_-_Imagine.m3u
./500_Greatest_Singles/singles.txt
./Blues/Muddy_Waters/Best_Of.m3u
./Blues/Robert_Johnson/Hellhound_On_My_Trail.MP3
./Blues/Johnny_Winter/Johnny_Winter.m3u
[Информация сокращена для экономии места]
```

Мы получили ответ, или, вернее, сведения, позволяющие его сформулировать. У нас есть файлы m3u, текстовые файлы, файлы, оканчивающиеся на .MP3 вместо .mp3. Кроме того, на диске содержатся изображения JPEG и GIF и даже видеофрагменты. Как и все программы Linux, команда find чувствительна к регистру символов (этот вопрос обсуждался в главе 1), поэтому при поиске по "*mp3*" не будут найдены файлы с суффиксом .MP3. Существует ли простой способ изменить имена файлов так, чтобы их суффиксы были представлены в нижнем регистре? Такой способ есть, и он предполагает использование команды find.

Выполнение действий над каждым найденным файлом

```
find -exec
```

Теперь вы готовы ознакомиться с той областью, в которой команда find реально проявляет свои огромные возможности. Над каждым файлом, найденным посредством команды find, можно выполнять произвольные действия.

После опций, "сужающих" поиск (например, -name, -type или -user), можно задать опцию -exec, а затем ввести команду, которую необходимо применить к файлам. Для представления каждого файла используются символы {}, а команду надо завершить обратной косой чертой, чтобы отменить специальное действие точки с запятой, иначе оболочка воспримет ее как разделитель при объединении команд (см. главу 4).

В предыдущем разделе мы выяснили, что имена некоторых файлов на диске оканчиваются последовательностью символов MP3. Поскольку мы предпочитаем суффиксы, составленные из символов нижнего регистра, преобразуем все вхождения MP3 в mp3. Для этой цели используем опцию -exec команды find. Сначала убедимся, что файлы с суффиксом .MP3 действительно существуют.

```
$ find . -name "Robert_Johnson*MP3"
./Blues/Robert_Johnson/Judgment_Day.MP3
./Blues/Robert_Johnson/Dust_My_Broom.MP3
./Blues/Robert_Johnson/Hellhound_On_My_Trail.MP3
```

Затем сформируем команду для изменения суффиксов. В качестве значения опции -exec зададим программу rename, которая позволяет изменять имя файла или его часть.

```
$ find . -name " *MP3" -exec
  rename 's/MP3/mp3/g' {} \;
```

Команде rename передаются инструкции по изменению имени в формате s/*предыдущий_компонент*/*новый_компонент*/g. (В данном случае s — первая буква слова substitute, а g означает global.) Ознакомимся с результатами выполнения команды.

```
$ find . -name "Robert_Johnson*MP3"
$ ls -1 Blues/Robert_Johnson/
```

```
Hellhound_On_My_Trail.mp3
Judgment_Day.mp3
Dust_My_Broom.mp3
Love_in_Vain.mp3
Me_and_the_Devil_Blues.mp3
```

Имена файлов преобразованы правильно. Применим тот же подход для решения другой задачи. В предыдущем разделе мы выяснили, что на диске содержатся файлы m3u. Так получилось, что в именах многих из них содержатся пробелы, что затрудняет работу с файлами. Попытаемся устранить этот недостаток. Сгенерируем список файлов m3u, содержащих пробелы. Используем шаблон поиска * *m3u. В нем слева и справа от пробела помещены символы групповой операции, т.е. таким образом выявляются файлы, в имени которых содержится хотя бы один пробел.

```
$ find . -name "* *m3u"
./Christmas/Christmas With The Rat Pack.m3u
./Christmas/Holiday_Boots_4_Your_Stockings.m3u
./Classical_Baroque/Handel/Chamber Music.m3u
./Classical_Opera/Famous Arias.m3u
./Doo_Wop/Doo Wop Box.m3u
./Electronica/Aphex_Twin/I Care Because You Do.m3u
```

Теперь к каждому найденному файлу применим команду rename. Символ для подстановки имеет вид "\ "; при этом команда rename получает информацию о пробеле, а оболочка не обрабатывает его нежелательным для нас образом. Пробел заменяем знаком подчеркивания.

```
$ find . -name "* *m3u" -exec rename 's/\ /_/g' {}
\;
$ find . -name "* *m3u"
$
```

Команда работает так, как мы и ожидали.

На заметку

Обратите внимание, что перед тем, как применять к файлам команду изменения имени, необходимо выяснить, какие файлы затронет эта операция. Не стоит пренебрегать такой проверкой, в противном случае вы можете получить нежелательные результаты.

Вывод результатов поиска в файл

```
find -fprint
```

Опцию `-print` задавать не обязательно. Она предполагается по умолчанию, следовательно, действовала во всех примерах, рассмотренных в данной главе. Если вы хотите вывести данные не на терминал, а в файл, вы должны задать опцию `-fprint`, указав после нее имя файла, который необходимо создать.

```
$ find . ! \( -name "*mp*" -o -name "*ogg" -o
-name "*flac" -o -type d \) -fprint
non_music_files.txt
$ cat non_music_files.txt
./Folk/Joan_Baez/Joan_Baez_-_Imagine.m3u
./500_Greatest_Singles/singles.txt
./Blues/Muddy_Waters/Best_Of.m3u
./Blues/Robert_Johnson/Hellhound_On_My_Trail.MP3
./Blues/Johnny_Winter/Johnny_Winter.m3u
```

Результаты поиска можно использовать при работе сценария, или, если, необходимо, вывести их на печать.

Выводы

Как вы уже знаете, с помощью команды `find` можно выполнять самые разнообразные действия, связанные с поиском информации. Возможности данной команды огромны.

Вы можете получить о них подробную информацию, выполнив команду man find или прочитав руководства, доступные в Web. Разнообразные опции find можно использовать для получения списка файлов и каталогов, а опция -exec делает данную программу поистине бесценной, позволяя применять произвольные команды к результатам поиска. Можно также передавать выходные данные find другим командам. Команда find — одна из самых полезных в системе Linux. Используйте ее!

11

Оболочка

До сих пор в данной книге речь шла о выполнении команд в оболочке, но не уделялось внимания самой оболочке. В этой главе мы рассмотрим две команды, оказывающие влияние на работу с оболочкой: `history`, отображающая список команд, введенных ранее в командной строке, и `alias`, которая создает псевдонимы для других команд. Обе эти команды часто используются и позволяют сэкономить много времени при работе с командной строкой. Лень — положительное качество пользователя, а две команды, о которых пойдет речь, дают возможность проявить свою лень в полной мере.

Просмотр списка предыстории

```
history
```

Каждый раз, когда вы вводите в оболочке очередную команду, она сохраняется в файле `.bash_history`, расположенном в вашем рабочем каталоге (точка в начале имени указывает на то, что файл скрытый, и, чтобы увидеть его, надо использовать команду `ls -a`). По умолчанию в файле сохраняется 500 строк. Если вы хотите увидеть список команд, введенных вами ранее, надо выполнить команду `history`.

```
$ history
496 ls
497 cd rsync_ssh
498 ls
499 cat linux
500 exit
```

Поскольку в файле содержится 500 строк, они быстро промелькнут на экране и вы не сможете рассмотреть их. Для того чтобы организовать поэкранный вывод, надо использовать уже известную вам конструкцию

```
$ history | less
```

Теперь просматривать данные стало намного удобнее.

Внимание!

Теперь вы понимаете, почему не следует вводить пароль и другие важные данные в командной строке. Каждый, кто может просмотреть содержимое файла `.bash_history`, получит доступ к ним.

Повторное выполнение последней команды

```
!!
```

Если вам надо снова выполнить команду, которая была вызвана последней, введите два восклицательных знака. В результате произойдет обращение к списку предыстории и будет вызвана команда, расположенная последней в списке.

```
$ pwd
/home/scott
$ !!
pwd
/home/scott
```

Заметьте, что на экране сначала отображается сама команда, а затем результаты ее выполнения. Этот способ позволяет вам избавиться от части рутинной работы, которую приходится выполнять при пользовании системой.

Вызов предыдущей команды путем указания ее номера

```
![##]
```

Команда `history` автоматически помещает перед именем каждой предыдущей команды номер. Если вы хотите вызвать одну из команд, выполненных ранее, и знаете ее номер в списке предыстории, то достаточно ввести восклицательный знак и указать после него номер команды. В результате команда будет выполнена повторно.

```
$ pwd
/home/scott
$ whoami
scott
$ !499
pwd
/home/scott
```

Если вы не помните номер, вам надо снова вызвать команду `history`. Заметьте, что в приведенном примере команда `pwd` сначала имела номер 499, но после того, как она будет вызвана с помощью выражения `!499`, ее номер изменится на 498, поскольку содержимое списка будет сдвинуто в результате включения в нее новой команды.

Вызов предыдущей команды путем указания строки символов

```
![строка]
```

Возможность вызова команды путем указания ее номера удобна, но для этого надо запомнить номера команд в списке предыстории, что достаточно трудно (можно, конечно, обработать выходные данные команды `history` посредством программы `grep`, но такое решение нельзя назвать оптимальным). Проще ссылаться на команду, выполненную ранее, по имени этой команды. Если вы введете после восклицательного знака несколько первых букв из имени команды, оболочка вызовет первую из подходящих команд, содержащихся в списке предыстории (просмотр `.bash_history` осуществляется от конца к началу).

```
$ cat /home/scott/todo
Buy milk
Buy dog food
Renew Linux Magazine subscription
$ cd /home/scott/pictures
$ !cat
cat /home/scott/todo
Buy milk
Buy dog food
Renew Linux Magazine subscription
```

Предположим, что в списке предыстории команда `cat` присутствует трижды: 35 (`cat /home/scott/todo`), 412 (`cat /etc/apt/sources.list`) и 496 (`cat /home/scott/todo`). При вводе `!cat` будет запущена команда с номером 496. Если вы хотите выполнить команду, имеющую номер 412, вам надо либо ввести `!412`, либо после восклицательного знака ввести информацию, достаточную для распознавания требуемой команды.

```
$ !cat /etc
cat /etc/apt/sources.list
deb http://us.archive.ubuntu.com/ubuntu breezy
  main restricted
deb-src http://us.archive.ubuntu.com/ubuntu breezy
  main restricted
```

Поскольку запомнить слово значительно проще, чем число, вы, вероятнее всего, предпочтете описанный здесь метод повторного вызова предыдущих команд. К тому же в любой момент вы можете вызвать команду `history` и ознакомиться со списком предыстории.

Отображение псевдонимов команд

```
alias
```

Если вы часто используете какую-либо команду, достаточно длинную и трудную для ввода, целесообразно создать для нее псевдоним. После этого, указав псевдоним, вы выполните ту команду, для которой он был создан. Конечно, если команда слишком сложная или занимает несколько строк, желательно оформить ее в виде сценария или функции. Но для относительно небольших команд псевдонимы — вполне приемлемое решение.

Псевдонимы хранятся в файле, находящемся в рабочем каталоге каждого пользователя. Имя этого файла `.bashrc` либо (что вероятнее) `.bash_aliases`. В большинстве дистрибутивных версий Linux уже определены некоторые псевдонимы. Для того чтобы просмотреть их список, надо ввести в командной строке команду `alias`.

```
$ alias
alias la='ls -a'
alias ll='ls -l'
```

Число предопределенных псевдонимов обычно невелико. При желании можно задать новые. Как это сделать, вы узнаете далее в этой главе.

Просмотр псевдонима конкретной команды

```
alias [имя псевдонима]
```

Определив несколько псевдонимов, легко забыть, какой из них принадлежит какой команде. По команде `alias` может отображаться слишком много информации, и, для того, чтобы разобраться в ней, может потребоваться время. Если вас интересует, какие действия будут выполнены при указании конкретного псевдонима, введите его имя после команды `alias`.

```
$ alias wgetpage
alias wgetpage='wget --html-extension -recursive
  --convert-links --page-requisites --no-parent $1'
```

В данном случае вы быстро и без труда получили сведения о том, какой команде соответствует псевдоним `wgetpage`.

На заметку

Программа `wget` будет подробно рассмотрена в главе 15.

Создание нового временного псевдонима

```
alias [псевдоним]='[команда]'
```

Если вы обнаружили, что вам слишком часто приходится вводить одну и ту же команду, пора определить для нее псевдоним. Так, для просмотра подкаталогов текущего каталога используется команда `ls -d */`. Для нее можно создать временный псевдоним следующим образом:

```
$ ls -d */
by_pool/   libby_pix/   on_floor/
$ alias lsd='ls -d */'
$ lsd
by_pool/   libby_pix/   on_floor/
```

Используя таким образом механизм псевдонимов, необходимо учитывать некоторые особенности. В имени псевдонима не должен присутствовать символ =, так как справа от него указывается команда, которой соответствует псевдоним. Сама команда может содержать этот символ. Кроме того, псевдоним, созданный таким способом действует лишь до завершения сеанса работы с оболочкой.

Если вы хотите создать псевдоним, который будет действовать и в последующих сеансах, вы сможете это сделать, прочитав следующий раздел.

Создание нового постоянно действующего псевдонима

```
alias [имя псевдонима]='[команда]'
```

Если вы хотите, чтобы созданный вами псевдоним не удалялся по окончании сеанса, вам надо включить его в файл, который оболочка использует для хранения псевдонимов. Обычно это либо `.bashrc`, либо `.bash_aliases`. Предположим, что в вашей системе используется `.bash_aliases`. Независимо от того, с каким файлом приходится работать, редактируя его, будьте внимательны, иначе при следующей регистрации могут возникнуть проблемы. А лучше всего перед редактированием файла создать его резервную копию.

На заметку

Как определить, какой файл следует использовать? Для этого достаточно ввести выражение `ls -a ~`. Если вы увидите имя `.bash_aliases`, используйте этот файл, в противном случае найдите файл `.bashrc` и проверьте, определены ли в нем другие каталоги. Если в этом файле нет определений псевдонимов, попробуйте найти файл `.profile`. В некоторых случаях используется именно он.

Для того чтобы включить псевдоним в файл `.bash_aliases`, откройте файл с помощью текстового редактора и добавьте строку наподобие следующей:

```
alias lsd='ls -d */'
```

То же правило, которое использовалось при создании временных псевдонимов, применимо и здесь: псевдоним не должен иметь знака равенства. После того как вы включили псевдоним в `.bash_aliases`, сохраните файл и закройте редактор. Вы увидите, что псевдоним еще не работает. Чтобы вновь созданный псевдоним вступил в действие, файл `.bash_aliases` (равно как и `.bashrc`) требует перезагрузки. Сделать это можно двумя способами. Вы можете либо завершить сеанс работы, а затем снова зарегистрироваться, что неудобно и поэтому не рекомендуется, либо ограничиться перезагрузкой файла с помощью следующей команды:

```
$ . .bash_aliases
```

Команда представляет собой точку, за которой следует пробел, а за ним — имя файла, начинающееся с точки. Теперь новый псевдоним должен работать. Поскольку вам придется перезагружать файл при каждом добавлении псевдонима, лучше определять сразу несколько псевдонимов; этим вы уменьшите объем рутинной работы, которую приходится выполнять.

Удаление всех псевдонимов

```
unalias
```

Задачи, которые приходится выполнять, работая на компьютере, меняются и может случиться так, что псевдоним будет больше не нужен. Для того чтобы удалить его используется команда unalias.

```
$ ls -d */
by_pool/ libby_pix/ on_floor/
$ alias lsd='ls -d */'
$ lsd
by_pool/ libby_pix/ on_floor/
$ unalias lsd
$ lsd
$
```

Заметьте, что данная команда позволяет безвозвратно удалить лишь временные псевдонимы, например псевдоним lsd из предыдущего примера. Если же вы примените эту команду к псевдониму, определенному в файле .bash_aliases, он также будет недоступен, но лишь в течение текущего сеанса. Когда вы повторно зарегистрируетесь в системе, его снова можно будет использовать.

Для того чтобы исключить псевдоним из файла .bash_aliases, надо открыть этот файл с помощью редактора и вручную удалить соответствующую строку. Если же вы считаете, что псевдоним может потребоваться вам в дальнейшем, ограничьтесь включением в начало этой строки символа #, как показано в следующем примере:

```
# alias lsd='ls -d */'
```

Сохраните файл .bash_aliases, перезагрузите его с помощью команды . .bash_aliases, и псевдоним не будет работать. Если у вас снова возникнет необходимость

в данном псевдониме, откройте файл `.bash_aliases`, удалите символ # и снова загрузите файл.

Выводы

Вы, как пользователь системы Linux, заинтересованы в том, чтобы, решая конкретную задачу, вводить с клавиатуры как можно меньше символов. В этом наверняка были убеждены авторы Unix, именно потому они назвали команду `ls`, а не `list`, `mkdir`, а не `makedirectory`. Команды `history` и `alias`, которые были рассмотрены в этой главе, позволяют сократить объем работы по вводу команд. Если вы устали вводить сложную и длинную команду, найдите ссылку на нее в списке предыстории или создайте псевдоним. В любом случае вы сэкономите время и усилия, да и клавиатура прослужит дольше.

12

Контроль использования системных ресурсов

Хороший пользователь должен быть немного системным администратором. Это значит, что ему следует проверять состояние машины, чтобы убедиться, что она работает корректно. Для того чтобы способствовать этой деятельности, в Linux предусмотрено несколько команд, предназначенных для отслеживания системных ресурсов. В данной главе мы рассмотрим несколько таких команд. Многие из них предназначены для выполнения различных действий, но для каждой из них есть задача, которую она решает лучше других (в этом авторы данных инструментов следуют лучшим традициям Unix — создавать небольшие программы, ориентированные на отдельные задачи). Диапазон задач, решаемых инструментами мониторинга, простирается от контроля программ, выполняющихся на компьютере (`ps` и `top`), до удаления нежелательных процессов (`kill`), вывода списка всех открытых файлов (`lsof`) и предоставления сведений об использовании оперативной памяти (`free`) и дискового пространства (`df` и `du`). Хорошо зная эти несложные инструменты, вы сможете предотвратить возникновение серьезных проблем.

Вывод информации о процессах, выполняемых в системе

```
ps aux
```

Иногда бывает, что запущенная вами программа "зависает" и перестает реагировать на ваши действия. При этом необходимо найти способ закрыть ее. Иногда приходится выяснять, какие из программ, запущенных некоторым пользователем, присутствуют в системе. В ряде случаев вам надо знать о всех процессах, выполняемых на машине в текущий момент. В каждой из этих ситуаций (а также в множестве других) надо использовать команду ps, отображающую информацию о процессах на вашем компьютере.

К сожалению, на данный момент существуют различные версии ps, поддерживающие разные типы опций. Более того, действие некоторых опций зависит от наличия или отсутствия дефиса перед ней, так что u и -u приводят к различным результатам. До сих пор все рассмотренные опции предварялись дефисом, однако, говоря о команде ps, нам придется отступить от этого правила.

Для того чтобы отобразить информацию о всех процессах, запущенных в системе любыми пользователями, надо задать после ps следующие опции: a (что означает all users, или все пользователи), u (user-oriented, или ориентированная на пользователя, т.е. команда должна отображать информацию о пользователе-владельце процесса) и x (processes without controlling ttys — процессы, не контролируемые ttys; еще одна категория процессов, необходимость отображения которых приходится задавать явно). Приготовьтесь получить длинный список процессов; в следующем примере он насчитывает 132 строки:

```
$ ps aux
USER  PID   %CPU %MEM VSZ     RSS    TTY   STAT
  START TIME   COMMAND
scott 24224 4.1  4.2  1150440 44036  ?     R
  11:02 12:03  /home/scott/.cxoffice/bin/wine-
preloader
scott 5594  0.0  0.1  3432    1820   pts/6 S+
  12:14 0:00   ssh scott@humbug.machine.com
scott 14957 0.3  7.5  171144  78628  ?     Sl
  13:01 0:35   /usr/lib/openoffice2/program/
  soffice.bin-writer
scott 12369 0.0  0.0  0       0      ?     Z
  15:43 0:00   [wine-preloader] <defunct>
scott 14680 0.0  0.1  3860    1044   pts/5 R+
  15:55 0:00   ps aux
```

Команда ps предоставляет подробную информацию: отображает сведения о пользователе-владельце процесса, идентификатор процесса (PID — Process ID), загрузку центрального процессора (%CPU) и использование памяти (%MEM) в процентах, текущее состояние процесса (STAT), а также имя процесса.

В столбце STAT содержится несколько различных букв. Наиболее важные из них приведены ниже.

Буква в столбце STAT	Значение
R	Выполняющийся
S	Спящий
T	Остановленный
Z	Зомби

Появление буквы Z — плохая новость. Она означает, что процесс "завис" и его нельзя завершить. Если у вас возникла проблема с программой и ps отображает для нее

состояние Z, вам, вероятно, придется перезагрузить машину, чтобы избавиться от этой программы.

Поскольку команда `ps aux` предоставляет большой объем данных, бывает трудно найти в списке информацию о конкретной программе. Передавая выходные данные `ps aux` программе `grep` для обработки, можно достаточно просто ограничить полученные результаты конкретной командой.

```
$ ps aux | grep [f]irefox
scott  25205  0.0  0.0  4336   8      ?  S
  Feb08 0:00   /bin/sh /opt/firefox/firefox
scott  25213  1.1  10.9 189092 113272 ?  Rl
  Feb08 29:42  /opt/firefox/firefox-bin
```

На данный момент мы имеем сведения об экземплярах Firefox, имеющихся в компьютере, а именно: пользователь, запустивший программу, какую нагрузку на компьютер создает эта программа и как долго она выполняется. Эта информация может оказаться очень полезной.

Совет

Почему мы выполнили поиск `[f]irefox`, а не `firefox`? Ответ на этот вопрос вы найдете в главе 9.

Просмотр дерева процессов

```
ps axjf
```

В системе Linux все процессы так или иначе взаимосвязаны. Некоторые программы в процессе выполнения запускают другие. Общим предком для всех процессов в системе является `init`, идентификатор которого всегда равен 1. Команда `ps` выводит текстовое представление дерева процессов, и вы можете выяснить, каким процессом был по-

рожден тот или иной процесс. Для отображения дерева процессов предназначены опции a и x, которые уже встречались в предыдущем разделе, а также опции j (job control) и f (forest).

На заметку

Как правило, программа ps отображает следующие столбцы:

```
PPID PID PGID SID TTY TPGID STAT UID TIME COMMAND
```

Для того чтобы упростить восприятие дерева команд, в листингах некоторые столбцы не приводятся.

```
$ ps axuf
  PPID   PID COMMAND
     1  7558 /usr/sbin/gdm
  7558  7561  \_ /usr/sbin/gdm
  7561  7604      \_ /usr/X11R6/bin/X :0
  7561  8225      \_ /bin/sh /usr/bin/startkde
  8225  8279          \_ /usr/bin/ssh-agent
  8225  8341          \_ kwrapper ksmserver
     1  8316 kdeinit Running...
  8316 10842 \_ konqueror [kdeinit] --silent
  8316 29663 \_ quanta
  8316 30906 \_ /usr/bin/kwrite /home/scott/analysis
  8316 17893 \_ /usr/lib/opera/9.0-20060206.1/opera
 17893 17960 |   \_ /usr/lib/opera/pluginwrapper
 17893 17961 |   \_ /usr/lib/opera/plugincleaner
```

Заметьте, что набор опций axjf приводит к появлению нового столбца PPID, представляющего идентификатор родительского процесса — процесса, породившего данный. Зная значения PID и PPID, вы сможете завершить процесс, вышедший из под контроля. Этот вопрос будет рассматриваться далее в данной главе.

Отображение процессов, принадлежащих конкретному пользователю

```
ps U [имя пользователя]
```

До сих пор мы рассматривали программу ps в режиме отображения списка всех процессов в системе. Если вы хотите ограничить набор результатов процессами, принадлежащими одному пользователю, вам надо задать опцию U и ввести после нее имя или числовой идентификатор этого пользователя.

```
$ ps U scott
PID    TTY      STAT   TIME
  COMMAND
14928  ?        S      0:00
  /opt/ooo2/program/soffice-writer
14957  ?        Sl     0:42
  /opt/ooo2/program/soffice.bin-writer
4688   pts/4    S+     0:00
  ssh scott@humbug.machine.com
26751  ?        Z      0:00
  [wine-preloader] <defunct>
27955  pts/5    R+     0:00
  ps U scott
```

Как видите, ps не включает в состав выходных данных пользовательское имя, но оно фигурирует в команде.

Совет

Если вам неизвестен числовой идентификатор пользователя, вы можете выяснить его, обратившись к файлу /etc/passwd. Достаточно найти пользовательское имя и прочитать в третьем столбце числовой идентификатор.

Завершение выполняющегося процесса

```
kill
```

Иногда программа начинает работать некорректно и не реагирует на попытки завершить ее обычным способом. Если эта программа предоставляет графический пользовательский интерфейс, то для ее завершения предусматривается специальная кнопка, а программа, выполняемая в режиме командной строки, должна завершаться после нажатия комбинации клавиш <Ctrl+C). Если программа выходит из под контроля, соответствующие действия пользователя не дают никакого результата. В этом случае необходимо применить команду `kill`.

С помощью программы `kill` процессу можно передать различные сигналы, от "завершить работу, по возможности приведя в порядок ресурсы", до "завершить работу немедленно". Рассмотрим три наиболее важных сигнала. Вызывая `kill`, вы можете задать "интенсивность" завершения процесса, указав одно из числовых или символьных значений, приведенных в табл. 12.1.

Таблица 12.1. Сигналы, передаваемые процессу посредством команды `kill`

Номер сигнала	Символьное обозначение	Действие
-1	-HUP	Процесс должен быть завершен. Прекратить его выполнение. (В применении к системным службам он означает перегрузку конфигурационных файлов и перезапуск соответствующей программы.)

Окончание табл. 12.1

Номер сигнала	Символьное обозначение	Действие
-15	-TERM	"Мягкое завершение" с удалением порожденных процессов и закрытием файлов
-9	-KILL	Прекратить все выполняющиеся действия и завершить работу

Обычно следует сначала попытаться установить значение -15 (если вы не зададите опции при вызове `kill`, данное значение будет принято по умолчанию). Этим вы даете программе шанс завершить зависимые процессы, закрыть файлы и выполнить другие действия по освобождению ресурсов. Если вы выждали некоторое время (какое именно — зависит от того, насколько вы терпеливы) и процесс по-прежнему выполняется и неуправляем или не отвечает, надо привести в действие "тяжелую артиллерию" и задать значение -9. Этим вы "выбиваете почву из под ног" работающей программы; она вынуждена бросить решаемую задачу в текущем состоянии, оставить порожденные процессы выполняющимися, а используемые файлы открытыми. Такой режим завершения нежелателен, но в некоторых случаях нет другого выхода.

Значение -1, или -HUP, предназначено в основном для служб, подобных Samba. Вы, вероятнее всего, будете использовать его достаточно редко, но должны знать, какие действия при этом выполняются.

Предположим, например, что процесс `grim` "завис". (Обычно он выполняется без проблем, но это лишь пример.)

```
$ ps U scott
  PID  TTY      STAT   TIME
  COMMAND
```

```
14928  ?       S      0:00
  /opt/ooo2/program/soffice-writer
14957  ?       Sl     0:42
  /opt/ooo2/program/soffice.bin-writer
 4688  pts/4   S+     0:00
  ssh scott@humbug.machine.com
26751  ?       Z      0:00
  [wine-preloader] <defunct>
27921  ?       Ss     0:00
  /usr/bin/gvim
27955  pts/5   R+     0:00
  ps U scott
$ kill 27921
$ ps U scott
PID    TTY     STAT   TIME
  COMMAND
14928  ?       S      0:00
  /opt/ooo2/program/soffice-writer
14957  ?       Sl     0:42
  /opt/ooo2/program/soffice.bin-writer
 4688  pts/4   S+     0:00
  ssh scott@humbug.machine.com
26751  ?       Z      0:00
  [wine-preloader] <defunct>
27955  pts/5   R+     0:00
  ps U scott
```

Для того чтобы завершить работу gvim, мы сначала должны выяснить его PID, вызвав команду ps. В данном случае значение идентификатора равно 27921. Далее следует передать PID программе kill, помня при этом, что по умолчанию она использует сигнал TERM, а затем снова проверить состояние процессов с помощью команды ps. Как видите, процесса gvim уже не существует.

На заметку

Почему мы не удалим с помощью команды `kill` процесс с идентификатором 26751, состояние которого обозначено буквой Z? Дело в том, что для зомби-процесса даже значение -9 не действует. Процесс уже "мертв", поэтому "убить" его невозможно. Единственный способ избавиться от него — перезагрузить систему.

Отображение динамически обновляемого списка выполняющихся процессов

```
top
```

Иногда пользователь обнаруживает, что быстродействие системы Linux без видимой причины внезапно уменьшилось. "Нечто" выполняется на компьютере настолько интенсивно, что занимает все процессорное время. Бывает, что, запустив программу, вы обнаруживаете, что она потребляет намного больше ресурсов центрального процессора, чем положено. Для того чтобы выяснить причины проблемы или хотя бы узнать, какие процессы выполняются в системе, можно вызвать программу `ps`, но после вывода данных она не обновляет их. Вы получаете лишь "мгновенный снимок" процессов в системе.

В отличие от `ps`, команда `top` представляет динамически обновляемые сведения о процессах и о том, какой объем системных ресурсов использует каждый из них. Работу команды `top` трудно проиллюстрировать в книге, так как листинги по своей природе статические. Можно лишь представить работу данной команды в один конкретный момент времени.

```
$ top
top - 18:05:03 up 4 days, 8:03, 1 user, load
  average: 0.83, 0.82, 0.97
```

```
Tasks: 135 total, 3 running, 126 sleeping,
  2 stopped, 4 zombie
Cpu(s): 22.9% us, 7.7% sy, 3.1% ni, 62.1% id,
  3.5% wa, 0.1% hi, 0.7% si
Mem: 1036136k total, 987996k used, 48140k free,
  27884k buffers
Swap: 1012084k total, 479284k used, 532800k free,
  179580k cached
  PID   USER    PR   NI   VIRT  RES   SHR  S  %CPU
  %MEM TIME+     COMMAND
25213  scott   15   0    230m  150m  15m  S  11.1
  14.9 33:39.34  firefox-bin
 7604  root    15   0    409m  285m  2896 S  10.8
  28.2 749:55.75 Xorg
 8378  scott   15   0    37084 10m   7456 S   1.0
  1.1 13:53.99  kicker
 8523  scott   15   0    69416 13m   3324 S   0.7
  1.3 63:35.14  skype
29663  scott   15   0    76896 13m   4048 S   0.7
  1.3 13:48.20  quanta
```

В первых пяти строках команда top отображает подробную информацию о системе, а затем описывает каждый выполняющийся процесс. Заметьте, что top автоматически сортирует выходные данные по уровню нагрузки на центральный процессор (%CPU), поэтому если интенсивность использования процессора программой изменится, то изменится также положение соответствующей строки в списке.

Если, работая с top, вы захотите удалить программу, вам достаточно ввести символ k. В начале списка, после строки, начинающейся со Swap:, вы увидите следующее сообщение:

```
PID to kill:
```

Введите идентификатор процесса, который вы собираетесь завершить (например, 8026), нажмите клавишу <Enter>, и программа запросит у вас номер сигнала.

```
Kill PID 8026 with signal [15]:
```

По умолчанию top использует сигнал с номером 15. Если это значение вам подходит, нажмите клавишу <Enter>, если нет — задайте номер сигнала и опять же завершите ввод нажатием клавиши <Enter>. Примерно через секунду информация о процессе исчезнет из списка, отображаемого программой top.

Для того чтобы завершить работу top, введите символ q.

Команда top чрезвычайно полезна, пользователи часто используют ее для того, чтобы выяснить, что случилось в системе. Если у вас возникнут вопросы, связанные с активностью программ, top, вероятнее всего, даст ответы на них.

Получение списка открытых файлов

```
lsof
```

В главе 1 говорилось о том, что любой объект на компьютере под управлением Linux — каталог, сетевое соединение, устройство — представляет собой файл. Это означает, что, хотя с вашей точки зрения открыт лишь один файл, на самом деле в системе в каждый момент присутствуют тысячи открытых файлов. Это не ошибка. Действительно тысячи. Для того чтобы увидеть полный список открытых файлов, надо использовать команду lsof (сокращение от list open files).

Если вы вызовете команду lsof без параметров (для этого вам надо выступать от имени пользователя root), то получите список, содержащий тысячи строк. В моей системе в данный момент открыто и используется 5497 фай-

лов. Таким образом, `lsof` предоставляет дополнительную информацию о состоянии компьютера в определенный момент времени.

Передавая выходные данные `lsof` программе `less`, можно организовать поэкранный просмотр результатов.

```
# lsof | less
COMMAND PID USER FD   TYPE DEVICE  SIZE  NODE
  NAME
init      1 root cwd DIR  3,1      656     2
  /
init      1 root rtd DIR  3,1      656     2
  /
init      1 root txt REG  3,1    31608  2072
  /sbin/init
```

Однако это не выход. Сколько страниц займут 5497 результатов? Для обработки выходных данных можно также использовать программу `grep`, но это не всегда нужно. Как вы узнаете из последующих разделов, сама программа `lsof` предоставляет возможность фильтрации данных так, что вы сможете сосредоточить внимание на том подмножестве открытых файлов, которое действительно интересует вас.

Отображение файлов, открытых конкретным пользователем

```
lsof -u
```

Если вы хотите получить список файлов, которые открыл конкретный пользователь (заметьте, что набор этих файлов включает также сетевые соединения и устройства), задайте при вызове команды `lsof` опцию `-u`, а затем введите пользовательское имя. (Помните, что сделать это вы сможете, лишь обладая правами `root`).

На заметку

Для экономии места некоторые данные, отображаемые командой `lsof`, не приводятся в этом и в последующих примерах:

```
# lsof -u scott
COMMAND     PID  USER   NAMEP
evolution  8603  scott  /home/scott/.evolution/
  addressbook/local/system/addressbook.db
opera     11638  scott  /usr/lib/opera/9.0/
  opera
opera     11638  scott  /usr/share/fonts/
  truetype/msttcorefonts/Arial_Bold.ttf
```

Даже когда мы ограничились информацией, соответствующей только одному пользователю, в списке все еще осталось 3039 строк. Некоторые пункты этого списка вызывают интерес. Во-первых, оказывается, что Evolution — программа обработки почты и персональной информации — постоянно выполняется. Кроме того, для отображения одной из страниц, отображаемых браузером Opera, требуется шрифт Arial Bold.

Если на компьютере, который вы администрируете, работают несколько пользователей, желательно просматривать информацию, касающуюся каждого из них, посредством команды `lsof -u`. Возможно, вы обнаружите, что они запускают программы, которые не должны использовать. Если вы — обычный пользователь системы Linux, применяйте эту команду для просмотра информации о своих файлах. Не исключено, что вы выявите программы, о наличии которых даже не догадывались.

Получение списка пользователей для конкретного файла

```
lsof [файл]
```

В предыдущем разделе вы узнали, как отображать сведения о файлах, открытых конкретным пользователем. Решим обратную задачу — выясним, кто использует конкретный файл. Для этого надо лишь ввести после `lsof` путь к этому файлу. Предположим, вы хотите узнать, кто использует демон SSH, предназначенный для удаленного взаимодействия с компьютером (опять же не забывайте, что команда `lsof` в этом режиме доступна только пользователю `root`).

```
# lsof /usr/sbin/sshd
COMMAND   PID USER   TYPE NAME
sshd     7814 root   REG  /usr/sbin/sshd
sshd    10542 root   REG  /usr/sbin/sshd
sshd    10548 scott  REG  /usr/sbin/sshd
```

Вы получили сведения о двух пользователях: `root` и `scott`. Если вы встретите неизвестное вам имя, например `4ackordood`, это означает, что система, вероятнее всего, подверглась атаке.

На заметку

С вашей точки зрения, `sshd` — это программа, но для системы Linux `/usr/sbin/sshd` — это всего лишь еще один файл.

Отображение сведений о процессах, соответствующих конкретной программе

```
lsof -c [программа]
```

В предыдущем разделе вы узнали, кто использует `/usr/sbin/sshd`. Однако полученные результаты не дают полной

картины происходящего. Каждая программа предполагает взаимодействие с несколькими (иногда очень многими) другими процессами, применение сокетов и устройств. Все они являются для системы Linux лишь файлами. Для получения более полных сведений о файлах, имеющих отношение к выполняемой программе, задайте при вызове команды `lsof` опцию `-c`, а после нее задайте имя выполняющейся (следовательно, "открытой") программы. Например, команда `lsof /usr/sbin/sshd` сообщает лишь о двух пользователях и трех открытых файлах. Но что можно сказать еще о команде `sshd`?

```
# lsof -c sshd
COMMAND   PID  USER NAME
sshd    10542  root  /lib/ld-2.3.5.so
sshd    10542  root  /dev/null
sshd    10542  root  192.168.0.170:ssh-
>192.168.0.100:4577 (ESTABLISHED)
sshd    10548  scott /usr/sbin/sshdp
sshd    10548  scott 192.168.0.170:ssh-
>192.168.0.100:4577 (ESTABLISHED)
```

В предыдущем листинге представлена лишь часть из 94 строк. Они представляют 94 открытых файла, так или иначе связанных с программой `sshd`. Среди них встречаются файлы `.so` (shared object — аналог DLL в Windows), а некоторые из файлов соответствуют сетевым соединениям между вашим компьютером и другой машиной (на самом деле удаленный компьютер посредством SSH обращается по сети к вашему (подробно этот вопрос рассмотрен в главе 15).

Применить команду `lsof` можно к самым разнообразным командам. Помимо получения полезной информации, вы узнаете, насколько сложными могут быть современные программы. Используйте `lsof` с программами,

которыми пользуетесь ежедневно, и вы оцените труд программистов, благодаря которым эти средства стали доступны вам.

На заметку

Команда `lsof` поддерживает большое число опций; мы рассмотрели лишь малую часть из них. В поставке исходного кода `lsof` содержится файл `00QUICKSTART` (имя начинается с двух нулей). Это руководство по наиболее важным применениям команды. Найдите с помощью Google этот файл и прочитайте его.

Отображение информации об оперативной памяти системы

```
free
```

Большинство современных компьютеров содержит сотни мегабайт и даже гигабайты оперативной памяти. Несмотря на это, нередко бывает, что работа компьютера замедляется вследствие слишком интенсивного использования памяти. Текущее состояние системной памяти позволяет выяснить команда `free`.

```
$ free
             total       used       free     shared    buffers     cached
Mem:       1036136     995852      40284          0      80816     332264
-/+ buffers/cache:     582772     453364
Swap:      1012084     495584     516500
```

По умолчанию команда `free` представляет результаты в килобайтах, так, как будто вы задали опцию `-k`. Однако ее поведение можно изменить. Опция `-b` задает отображение сведений о памяти в байтах, а опция `-m` (именно она используется чаще всего) — в мегабайтах.

```
$ free -m
             total       used       free     shared    buffers
cached
Mem:          1011        963         48          0         78        316
-/+ buffers/cache:        569        442
Swap:          988        483        504
```

Из выходных данных команды `free` мы видим, что на машине есть 1011 Мбайт доступной оперативной памяти (реально объем памяти равен 1024 Мбайт, но `free` отображает значение 1011, потому что остальные 13 Мбайт заняты ядром системы и недоступны для других пользователей). Далее мы видим, что 963 Мбайт памяти используется, а свободными остаются лишь 48 Мбайт. Также в системе насчитывается почти гигабайт виртуальной памяти и около половины ее, а именно 483 Мбайт, также заняты. Таково положение дел в данный момент.

Однако это еще не все. Очень важна информация в строке, начинающейся с `-/+ buffers/cache`. Для ускорения работы жестких дисков используются буферы и кэш. Если эта память понадобится, она может быть быстро освобождена для программ. С точки зрения приложений, выполняющихся на машине под управлением Linux, 442 Мбайт памяти свободны и могут использоваться, а 569 Мбайт можно освободить в случае необходимости. Это немалое дополнение к виртуальной памяти. Система Linux вряд ли была бы столь популярной, если бы не обеспечивала эффективного управления памятью.

Совет

Web-страница, содержащая более подробную информацию о команде `free` и управлении памятью в системе Linux, доступна по адресу `http://gentoo-wiki.com/FAQ_Linux_Memory_Management`. Несмотря на то что представленная на ней информация ориентирована на Gentoo (редко встречающаяся разновидность Linux), она применима и для других версий системы.

Отображение информации об использовании дискового пространства

```
df
```

Команда free предоставляет сведения об оперативной памяти, имеющейся в системе. Подобно ей, команда df (disk free) сообщает об объеме доступного дискового пространства. Вызовите команду df, и вы получите список дисков, данные о свободной области на них и смонтированных файловых системах.

```
$ df
Filesystem  1kblocks     Used  Available  Use%
  Mounted on
/dev/hda1    7678736  5170204    2508532   68%
  /
tmpfs         518068        0     518068    0%
  /dev/shm
tmpfs         518068    12588     505480    3%
  /lib/modules/2.6.12-10-386/volatile
/dev/hda2   30369948 24792784    5577164   82%
  /home
```

Перед тем как подробно рассматривать результаты выполнения команды, обсудим их структуру. По умолчанию команда df выводит данные в килобайтах, однако они становятся проще для восприятия, если вы используете опцию -h (или --human-readable).

```
$ df -h
Filesystem   Size   Used   Avail   Use%
  Mounted on
/dev/hda1    7.4G   5.0G    2.4G    68%
  /
tmpfs        506M      0    506M     0%
  /dev/shm
```

```
tmpfs           506M     13M     494M      3%
  /lib/modules/2.6.12-10-386/volatile
/dev/hda2        29G     24G      5.4G     82%
  /home
```

Термин "human-readable" означает, что килобайты представляются буквой K, мегабайты — M, а гибатайты — G. Буквы M и G присутствовали в последнем листинге.

Что же означают эти результаты? В данном случае на жестком диске есть два раздела: /dev/hda1, смонтированный в каталоге /, и /dev/hda2, смонтированный в разделе /home. Раздел /home имеет объем 29 Гбайт, 82% которого используется в данный момент. Свободным остается около 5,4 Гбайт. Пока еще нет оснований для беспокойства, но если в данный раздел будет записано содержимое еще нескольких компакт-дисков, придется подумать о том, чтобы удалить ненужные файлы.

В корневом разделе, или /, объем свободного пространства несколько меньше: 2,4 Гбайт из 7,4 Гбайт. Этот раздел вряд ли будет быстро заполняться, поскольку в нем содержатся программы и другие относительно статические файлы. В нем вы найдете каталог /var, в котором размещаются данные, используемые при инсталляции (процесс инсталляции будет описан в главе 13), а также файлы, размер и содержимое которых постоянно изменяются (например, файлы протоколов). Несмотря на это, общий объем данных в каталоге изменится мало, если, конечно, вы не планируете устанавливать сложные приложения или разместить на данной машине Web-сервер или сервер баз данных.

Остальные два раздела помечены как tmpfs, это означает, что они содержат временные файловые системы, используемые виртуальной памятью, или областью подкачки. При выключении компьютера данные в этих "разделах" теряются.

Совет

Дополнительную информацию о tmpfs можно найти в статье из Wikipedia по адресу http://en.wikipedia.org/wiki/TMPFS.

Определение размера области, занятой содержимым каталога

```
du
```

Команда df сообщает о состоянии всего жесткого диска. Но что делать, если вам надо узнать, какой объем дисковой памяти занимают каталог и файлы, содержащиеся в нем? Ответ на этот вопрос даст команда du (сокращение от disk usage). Сначала надо сделать текущим интересующий вас каталог, а затем выполнить команду du.

```
$ cd music
$ du
36582   ./Donald_Fagen
593985  ./Clash
145962  ./Hank_Mobley/1958_Peckin'_Time
128200  ./Hank_Mobley/1963_No_Room_For_Squares
108445  ./Hank_Mobley/1961_Workout
2662185 .
```

Как и в случае команды df, результаты, сгенерированные du, представляются в килобайтах, и она также поддерживает опцию -h (или --human-readable), позволяющую вывести данные в формате, более удобном для восприятия.

```
$ cd music
$ du -h
36M   ./Donald_Fagen
581M  ./Clash
```

```
143M ./Hank_Mobley/1958_Peckin'_Time
126M ./Hank_Mobley/1963_No_Room_For_Squares
106M ./Hank_Mobley/1961_Workout
2.6G .
```

Полученные результаты позволяют определить объем дисковой памяти, занимаемой каждым подкаталогом. Обратите внимание на каталог `Hank_Mobley`, который занимает 374 Мбайт. Этот объем получен путем суммирования объема всех подкаталогов, содержащихся в составе `Hank_Mobley` (поскольку реальный объем в килобайтах при представлении в мегабайтах округлятся, значения объемов несколько различаются). Если объем `Hank_Mobley` существенно больше, чем объем трех подкаталогов, это значит, что каталог `Hank_Mobley` содержит обычные файлы, расположенные за пределами подкаталогов.

В конце списка приводится общий размер всего каталога `music/`. Он составляет 2,6 Гбайт.

Ограничение вывода общим размером пространства, занятого каталогом

```
du -s
```

Если вам не нужна информация о подкаталогах, вы можете задать `du`, чтобы вывести лишь общий объем. Для этого используется опция `-s`.

```
$ cd music
$ du -hs
2.6G
```

Результаты получаются достаточно компактными.

Выводы

Для каждой из команд, рассмотренных в данной главе, существует вариант, оснащенный графическим пользовательским интерфейсом. Однако если система не поддерживает графических средств, или если вы обращаетесь к машине посредством SSH (этот вопрос будет рассмотрен в главе 15) и не можете использовать графический интерфейс, или если вы хотите воспользоваться наиболее быстродействующим инструментом, вам придется работать с командной строкой. Благодаря удачно выбранным именам, команды, описанные в данном разделе, легко запомнить.

- `ps` (view running processes — просмотр информации о выполняющихся процессах);
- `kill` (завершение работы процессов);
- `top` (верхняя часть списка выполняющихся процессов);
- `lsof` (list [ls] open files — список открытых файлов);
- `free` (свободная, или доступная, память);
- `df` (disk free space — свободное место на диске);
- `du` (disk usage — информация об использовании дискового пространства).

Применяйте эти команды при работе на своем компьютере и не забывайте читать справочную информацию. В данной главе была рассмотрена лишь незначительная часть их возможностей. Команды `ps`, `top` и `lsof` поддерживают большое количество опций, что позволяет в деталях контролировать поведение компьютера, на котором вы работаете.

13

Инсталляция программного обеспечения

В состав дистрибутивных пакетов Linux входят тысячи программ. Сразу же после установки системы вы можете обращаться к Web-страницам из глобальной сети, подготавливать отчеты, создавать электронные таблицы, просматривать изображения, воспроизводить звуковые файлы и выполнять многие другие действия. Но несмотря на обилие программ, очень часто возникает необходимость в установке новых пакетов. К счастью, Linux позволяет без труда инсталлировать программное обеспечение на компьютер.

Раньше считалось, что любую программу, которая должна быть установлена в системе Linux, необходимо скомпилировать. В принципе никто не запрещает делать этого (так поступают многие), но в настоящее время реальная необходимость в компиляции возникает достаточно редко. Вы можете ограничиться инсталляцией пакетов, используя для этой цели простые инструменты.

Приступая к реальной установке программ, необходимо учесть следующее. В системе Linux используются различные форматы программных пакетов, но два из них, RPM и DEB, встречаются чаще других (именно их мы и рассмотрим в этой главе). RPM применяется в системах типа Red Hat (да и само название формата расшифровывается как

Red Hat Package Manager), Fedora Core, SUSE и некоторых других. DEB ориентирован на системы Debian, K/Ubuntu, Linspire, Xandros и т.д. Необходимо представлять себе работу обоих форматов, но ориентироваться, естественно, на системы, соответствующие вашей версии Linux.

Совет

Подробный анализ систем управления пакетами и версий Linux, использующих их, приведен в документе *Linux Distributions — Facts and Figures: What is your distribution's package management?*, доступном по адресу `http://distrowatch.com/stats.php?section=packagemanagement`. Заметьте, что лидирующим инструментом является DEB, который был разработан с использованием средств K/Ubuntu.

Инсталляция программных пакетов в RPM-системах

```
rpm -ihv [пакет]
rpm -uhv [пакет]
```

Команда `rpm` инсталлирует пакеты, имена которых заканчиваются символами `.rpm` (что вполне логично). Для установки RPM-пакета его сначала надо скопировать на локальный компьютер. Рассмотрим в качестве примера сканер портов `nmap`. RPM-пакет `nmap` можно скопировать, обратившись по адресу `http://www.insecure.org/nmap/download.html`. После того как пакет окажется в вашей системе, вам достаточно будет вызвать `rpm`, задав три опции: `-i` (install), `-h` (для отображения хода процесса инсталляции) и `-v` (verbose, т.е. вывод подробной информации о выполняемых действиях). Данная команда должна быть вызвана от имени пользователя `root`.

```
# rpm -ihv nmap-4.01-1.i386.rpm
```

Однако эту команду нельзя рекомендовать для использования. Гораздо лучше установить набор опций `-Uhv`, где `U` — сокращение от `upgrade`. Почему же `-U` лучше, чем `-i`? Дело в том, что `-i` задает только инсталляцию, а `-U` — обновление и инсталляцию. Если пакет уже инсталлирован в системе и вы хотите установить его новый вариант, опция `-U` лишь дополнит его до новой версии. Если же пакет в системе отсутствует, то опция `-U` приведет к его инсталляции. Таким образом, лучше всего задавать опцию `-U`; независимо от того, собираетесь ли вы обновить версию или инсталлировать программу, данная опция выполнит нужные действия.

```
# rpm -Uhv nmap-4.01-1.i386.rpm
Preparing...  ########################################### [100%]
1:nmap        ########################################### [100%]
```

Если вы хотите инсталлировать несколько RPM-пакетов, их имена надо разделить пробелами.

```
# rpm -Uhv nmap-4.01-1.i386.rpm nmap-frontend-4.01-
1.i386.rpm
```

Если же число пакетов велико, вы даже можете использовать символы групповых операций. Предположим, например, что в вашем распоряжении есть двадцать файлов `.rpm`, расположенных в подкаталоге `software`. Вы можете установить все пакеты с помощью следующей команды:

```
# rpm -Uhv software/*.rpm
```

Внимание!

Опции `-U` следует отдавать предпочтение всегда, за исключением установки ядра. В этом случае надо задавать опцию `-i`. Если вы выполните обновление с помощью `-U` и новое ядро не будет работать, вы столкнетесь с серьезными проблемами. С другой стороны, если вы зададите опцию `-i`, то старое ядро останется на вашей машине в качестве резервной копии. Если новое ядро не будет работать, вы сможете воспользоваться старым.

Удаление программных пакетов из RPM-систем

```
rpm -e [пакет]
```

Удалить инсталлированный пакет RPM даже проще, чем установить его. Вместо -Uhv надо задать опцию -e (erase).

```
# rpm -e nmap
```

Вот и все. Опция -e не отображает данных. Обратите внимание на то, что когда вы инсталлировали программу, используя опцию -Uhv, вам надо было указать имя файла, иначе rpm не смогла бы узнать, где находятся данные для инсталляции. При удалении программного обеспечения вы указываете имя пакета. Программа rpm распознает пакет по его имени, а не по имени файла, так как файлы, использованные для инсталляции, уже могли давно удалить.

Инсталляция зависимых программных пакетов в RPM-системах

```
yum install [пакет]
```

Команда rpm предоставляет большие возможности, но поработав с ней, вы рано или поздно столкнетесь с проблемой установки зависимых пакетов. Для того чтобы установить пакет A, вам также надо скопировать и инсталлировать пакеты B и C, но чтобы инсталлировать C, необходимо скопировать и установить пакеты D и E, а чтобы установить пакет E ... и т.д. В системе Debian и других подобных ей (о них пойдет речь далее в этой главе) данная проблема была решена давно путем использования программы apt. Эта программа может быть применена и в RPM-системах, но гораздо чаще используется сравнительно новый инструмент yum. Первоначально программа yum была разработана

для Yellow Dog Linux (отсюда имя, которое расшифровывается как Yellow Dog Updater, Modified). В настоящее время она широко используется, но пока уступает по возможностям программе apt. Поскольку yum доступна в RPM-системах, мы рассмотрим ее здесь.

Команда yum инсталлирует, обновляет и удаляет программные пакеты, действуя как оболочка для rpm, кроме того, она автоматически отслеживает зависимости. Например, если вы пытаетесь установить пакет A, о котором шла речь в начале данного раздела, yum скопирует и инсталлирует A, B и C. Если впоследствии вы решите, что пакет A вам не нужен, yum удалит не только его, но также B и C (при условии, что другие программные пакеты не используют их).

Инсталлировать программы с помощью yum очень просто. Предположим, что вы хотите установить программу воспроизведения звуковых файлов XMMS (ее имя расшифровывается как X Multimedia System). Для того чтобы установить XMMS, надо также инсталлировать несколько зависимых пакетов. Программа yum существенно упрощает этот процесс по сравнению с инсталляцией посредством rpm. Для того чтобы начать работу, вам не надо самостоятельно искать и копировать пакет xmms, программа yum сама скопирует не только его, но и все зависимые пакеты.

К сожалению, в процессе работы yum выводит слишком много информации. Приведенные ниже данные были существенно сокращены, и все равно объем листинга остается большим. Ознакомившись с этим листингом, вы можете составить представление о том, что вы увидите при работе с программой yum.

```
# yum install xmms
Setting up Install Process
Setting up repositories
update 100% |=========================| 951 B 00:00
```

```
base 100% |=========================| 1.1 kB 00:00

Resolving Dependencies
--> Populating transaction set with selected
packages. Please wait.
---> Downloading header for xmms to pack into
transaction set.
xmms-1.2.10-9.i386.rpm 100% |==========| 24 kB
00:00
--> Restarting Dependency Resolution with new
changes.

---> Downloading header for gtk+ to pack into
transaction set.
gtk%2B-1.2.10-33.i386.rpm 100% |==========|
23 kB 00:00

Dependencies Resolved

Package    Arch    Version        Repository    Size

Installing:
 xmms      i386    1:1.2.10-9     base          1.9 M
Installing for dependencies:

libogg     i386    2:1.1.2-1      base           16 k
libvorbis i386     1:1.1.0-1      base          185 k


Total download size: 3.3 M
Is this ok [y/N]:
```

После того как вы введете y, программа yum начнет копирование и инсталляцию пакетов, продолжая комментировать каждый шаг своей работы.

```
Downloading Packages:
...

(5/6): libogg-1.1.2-1.i38 100% |==========|
16 kB 00:00
(6/6): libvorbis-1.1.0-1. 100% |==========|
185 kB 00:00

Running Transaction Test

Running Transaction
  Installing: libogg      ########## [1/6]
  Installing: libvorbis   ########## [2/6]

...
Installed: xmms.i386 1:1.2.10-9
Dependency Installed: gdk-pixbuf.i386 1:0.22.0-
17.el4.3 gtk+.i386 1:1.2.10-33 libogg.i386
2:1.1.2-1 libvorbis.i386 1:1.1.0-1 mikmod.i386
0:3.1.6-32.EL4
```

Наконец инсталляция XMMS завершилась, и программа доступна для использования. Теперь рассмотрим, как в случае необходимости избавиться от XMMS.

Удаление зависимых программных пакетов из RPM-систем

```
yum remove [пакет]
```

Программа yum обладает важной положительной особенностью: ее система команд дружественна по отношению к пользователю. Хотите инсталлировать пакеты? Введите yum install. Хотите удалить его? Введите yum remove. Так, если вам надо избавиться от XMMS, задайте следующую команду:

```
# yum remove xmms
Setting up Remove Process
Resolving Dependencies

---> Package xmms.i386 1:1.2.10-9 set to be erased

Dependencies Resolved

  Package    Arch    Version      Repository     Size

Removing:
   xmms      i386    1:1.2.10-9   installed      5.2 M

Is this ok [y/N]:
```

Даже в таком простом деле, как удаление программного пакета, программа yum продолжает надоедать вам подробными комментариями. Введите y, чтобы подтвердить готовность к удалению программы, и вы увидите дополнительные сообщения.

```
Running Transaction Test

Running Transaction
 Removing : xmms          ################### [1/1]
Removed: xmms.i386 1:1.2.10-9
Complete!
```

Теперь программа XMMS удалена. Заметьте, что зависимые пакеты, инсталлированные с помощью программы yum, остались в неприкосновенности. Они нужны были для работы XMMS, но ими могут пользоваться другие программы на вашем компьютере, поэтому они остаются на месте (как вы увидите далее, по умолчанию такое же поведение демонстрирует программа apt).

Обновление программных пакетов в RPM-системах

```
yum update
```

Обычно в системе Linux установлены сотни, а то и тысячи программных пакетов. То один, то другой приходится постоянно обновлять. Если бы вам пришлось вручную отслеживать появление новых версий программ и устанавливать необходимые дополнения, это заняло бы все ваше время. К счастью, есть средства, позволяющие упростить этот процесс. Простая команда `yum update` говорит программе `yum` о том, что необходимо искать обновления к управляемым ею программам. Если новые пакеты доступны, программа `yum` оповещает вас о появившихся возможностях и запрашивает подтверждение на продолжение инсталляции.

```
# yum update
Setting up Update Process
Setting up repositories
update    100%  |=================|  951 B  00:00
base      100%  |=================| 1.1 kB  00:00

Resolving Dependencies
--> Populating transaction set with selected
packages. Please wait.
---> Downloading header for cups-libs to pack
into transaction set.
cups-libs-1.1.22-0.rc1.9. 100% |=========| 22 kB
00:00
---> Package cups-libs.i386 1:1.1.22-0.rc1.9.10
set to be updated
```

```
--> Running transaction check
Dependencies Resolved
Package  Arch  Version              Repository    Size

Installing:
openssl  i686  0.9.7a-43.4         update        1.1 M
pam      i386  0.77-66.13          update        1.8 M
perl     i386  3:5.8.5-24.RHEL4    update         11 M
udev     i386  039-10.10.EL4.3     update        830 k
wget     i386  1.10.2-0.40E        update        567 k
Transaction Summary

Install    1 Package(s)
Update    11 Package(s)
Remove     0 Package(s)
Total download size: 30 M
Is this ok [y/N]:
```

Если на данном этапе работы вы введете y, вы дадите этим согласие на копирование и инсталляцию двенадцати пакетов. После вывода пространных сообщений yum завершит свое выполнение, и ваш компьютер будет готов к дальнейшей работе. Хотите всегда быть на переднем крае? Запускайте программу yum ежедневно. Если вы не испытываете необходимости всегда иметь в своем распоряжении последние версии, запускайте программу yum реже, но делайте это регулярно. Дополнения, предназначенные для повышения уровня защиты, появляются очень часто, поэтому желательно иметь их на вооружении.

Поиск пакетов, готовых к копированию на RPM-системы

```
yum search [строка]
yum list available
```

Теперь вы знаете, как инсталлировать и удалять программное обеспечение с помощью программы yum, но как найти его? Предположим, что вас интересует пакет для работы с изображениями GIMP (GNU Image Manipulation Program). Вы хотите знать, есть ли пакеты, имеющие отношение к GIMP, готовые к инсталляции посредством программы yum. Можно выполнить команду `yum search gimp`, но это далеко не идеальное решение. Данная команда будет искать соответствие условиям поиска в именах всех пакетов, в аннотациях и даже в списках имен программы для работы с пакетами. В результате вы получите список, включающий чуть ли не все программы, известные в мире.

Лучшим решением будет запросить список пакетов, доступных посредством программы yum (размеры списка и в этом случае будут невероятно большими), а затем средствами конвейерной обработки передать результаты для обработки программе grep.

```
$ yum list available | grep gimp
gimp.i386             1:2.0.5-5 base
gimp-devel.i386       1:2.0.5-5 base
gimp-help.noarch      2-0.1.0.3 base
gimp-print.i386       4.2.7-2   base
```

В данном примере мы получили одиннадцать результатов — вполне приемлемое количество. Если вы действительно хотите выполнить полномасштабный поиск, применяйте команду `yum search`, в противном случае используйте `yum list available` и программу grep. В большинстве случаев второй способ оказывается наиболее приемлемым.

Инсталляция программных пакетов в Debian

```
dpkg -i [пакет]
```

Инсталляция нового программного обеспечения — одно из самых приятных занятий при работе в системе Linux. Как вы узнаете из последующих разделов, в системе Debian используется программа apt — самый мощный и простой в применении инструмент установки программ. Эта программа предоставляет богатые возможности, но большинство из них она реализует, выступая в роли оболочки вокруг программы dpkg (подобно тому, как yum является оболочкой для rpm). Программа dpkg выполняет черновую работу по инсталляции и удалению программ на машине под управлением Debian. Перед тем как рассматривать apt, научимся работать с dpkg, поскольку apt не обладает универсальным возможностями по инсталляции любых программ.

Рассмотрим следующий пример. В настоящее время одной из самых популярных программ VoIP (Voice over IP) является Skype. Однако лицензионные соглашения не позволяют включать Skype в состав дистрибутивного пакета операционной системы. Если вы хотите воспользоваться Skype, вам надо сначала скопировать эту программу с сервера компании, а затем установить ее вручную. Обратимся к странице с информацией о пакетах, доступных для копирования, по адресу `http://www.skype.com/products/skype/linux/` и найдем файл для нашей версии системы. В данном случае нам нужен пакет для Debian — `skype_1.2.0.18-1_i386.deb`.

После того как пакет будет скопирован в систему, надо инсталлировать его. Прежде всего с помощью команды cd сделайте текущим каталог, содержащий пакет, а затем инсталлируйте его посредством программы dpkg.

На заметку

В большинстве систем, подобных Debian, команды, предполагающие обращение к dpkg, должны выполняться с правами root. С другой стороны, в популярной версии K/Ubuntu данное правило не действует. Вместо этого команды предваряются sudo. Одним словом, в Debian используется следующая команда:

```
# dpkg -i skype_1.2.0.18-1_i386.deb
```

В K/Ubuntu и других sudo-системах выражение в командной строке должно иметь такой вид:

```
$ sudo dpkg -i skype_1.2.0.18-1_i386.deb
```

При написании этой книги использовались машины под управлением K/Ubuntu, поэтому, если вы увидите команду sudo, знайте, зачем она нужна.

```
# ls
skype_1.2.0.18-1_i386.deb
# dkpg -i skype_1.2.0.18-1_i386.deb
sudo dpkg -i skype_1.2.0.18-1_i386.deb
Selecting previously deselected package skype.
(Reading database ... 97963 files and directories
currently installed.)
Unpacking skype (from skype_1.2.0.18-1_i386.deb) ...
Setting up skype (1.2.0.18-1) ...
```

Вот и все. Команда dpkg отличается краткостью и сообщает вам только самые важные сведения и ничего сверх этого.

Удаление программных пакетов из системы Debian

```
dpkg -r [пакет]
```

Опция -i, использующаяся для инсталляции программного обеспечения на машины под управлением Debian,

означает *install*. Аналогично, опция -r, удаляющая программы, означает *remove*. Если вам больше не нужна программа Skype, вы можете легко удалить ее со своего компьютера.

```
# dpkg -r skype
(Reading database ... 98004 files and directories
currently installed.)
Removing skype ...
```

Когда вы инсталлировали программу, используя dpkg -i, вам надо было указать имя файла, иначе dpkg не смогла бы узнать, где находятся данные для инсталляции. При удалении программного обеспечения посредством команды dpkg -r надо ввести имя пакета. Программа dpkg распознает пакет по его имени, а не по имени файла, так как файлы, использованные для инсталляции, уже могли давно удалить.

Инсталляция зависимых пакетов в системе Debian

```
apt -get install [пакет]
```

Команда dkpg предоставляет большие возможности, но поработав с ней, вы рано или поздно столкнетесь с проблемой установки зависимых пакетов. Для того чтобы установить пакет A, вам также надо скопировать и инсталлировать пакеты B и C, но чтобы инсталлировать C, необходимо скопировать и установить пакеты D и E, а чтобы установить пакет E... и т.д. Вам нужна программа apt!

Команда apt, как и средства, рассмотренные ранее, позволяет инсталлировать, обновлять и удалять программные пакеты, кроме того, она автоматически обрабатывает зависимые пакеты. Например, если вы пытаетесь установить

пакет A, о котором шла речь в начале данного раздела, apt скопирует и инсталлирует A, B и C. Если впоследствии вы решите, что пакет A вам не нужен, apt удалит не только его, но также B и C (при условии, что другие программные пакеты не используют их).

Команда apt изначально была разработана для использования в системе Debian в качестве интерфейса к dpkg. В настоящее время она имеется в каждом дистрибутивном пакете на базе Debian — в самой системе Debian, в K/Ubuntu, Linspire, Xandros и во многих других. Это один из тех инструментов, благодаря которым система Debian столь мощна и проста в использовании. Специалисты, использующие системы, отличные от Debian, по достоинству оценивают преимущества команды apt и стараются применить ее для работы в RPM-системах. В данной главе мы сосредоточим внимание лишь на использовании команды apt с Debian.

Совет

Обзор вопросов использования apt в RPM-системах можно найти в статье *A Very Apropos apt*, опубликованной в октябрьском номере *Linux Magazine* за 2003 г. Текст статьи хранится по адресу `http://www.linux-mag.com/2003-10/apt_01.html`. Кроме того, новые сведения об RPM-системах можно найти по адресу `http://apt.freshrpms.net/`. Для доступа к содержимому Linux Magazine требуется регистрация, но эта услуга предоставляется бесплатно.

Предположим, вы хотите инсталлировать с помощью apt сложный инструмент sshfs. Чтобы сделать это, вам надо выполнить следующую последовательность команд (помните, что для вызова этих команд надо обладать правами root):

```
# apt-get update
Get:1 http://us.archive.ubuntu.com breezy
Release.gpg [189B]
```

```
Get:2 http://archive.ubuntu.com breezy
Release.gpg [189B]


Hit ftp://ftp.free.fr breezy/free Sources
Hit ftp://ftp.free.fr breezy/non-free Sources
Fetched 140kB in 1m4s (2176B/s)
Reading package lists... Done
[Results truncated for length]
# apt-get install sshfs
Reading package lists... Done
Building dependency tree... Done
The following extra packages will be installed:
  fuse-utils libfuse2
The following NEW packages will be installed:
  fuse-utils libfuse2 sshfs

Need to get 96.9kB of archives.
After unpacking 344kB of additional disk space
will be used.
Do you want to continue [Y/n]? y
Get:1 http://us.archive.ubuntu.com breezy/universe
sshfs 1.1-1 [19.3kB]
...
Fetched 96.9kB in 10s (9615B/s)
Reading package fields... Done
Reading package status... Done
Preconfiguring packages ...
...
Selecting previously deselected package sshfs.
Unpacking sshfs (from
.../archives/sshfs_1.1-1_i386.deb) ...

Setting up sshfs (1.1-1) ...
```

Рассмотрим подробно выполненные действия. В данном случае были вызваны две команды. По команде `apt-get update` произошла загрузка списка текущих программных пакетов с серверов (говоря об установке программ, принято использовать для их обозначения термин "хранилище"), указанных в конфигурационном файле `apt /etc/apt/sources.list`. (Если вы хотите узнать, что это за серверы, выполните команду `cat /etc/apt/sources.list`.) Если в начале строки, следующей за `apt-get update`, вы увидите слово `Get`, то это означает, что состояние хранилища изменилось и надо снова скопировать список. Слово `Ign` означает, что информация на компьютере соответствует состоянию хранилища и выполнять копирование не надо. Вызов `apt-get update` перед другими действиями позволяет убедиться, что список пакетов корректен и вовремя обновлен.

Команда `apt-get install sshfs` извлекает указанный пакет, а также пакеты, зависящие от него (в данном случае это `fuse-utils` и `libfuse2`). После того как они будут доставлены на ваш компьютер, команда `apt` (а реально `dpkg`) инсталлирует требуемое программное обеспечение. Помните, что вам следует всегда использовать имя пакета, а не имя файла. Другими словами, команда должна иметь вид `apt-get install sshfs`, а не `apt-get install sshfs_1.1-1_i386.deb`. Если `apt` выявит дополнительные зависимости для запрашиваемого пакета, как это имело место с `sshfs`, вам надо подтвердить, что вы согласны инсталлировать их.

Если вам надо инсталлировать одновременно несколько пакетов, укажите их в командной строке. Например, если вы хотите установить `sshfs` и `shfs-utils`, командная строка должна иметь следующий вид:

```
# apt-get install sshfs shfs-utils
```

При этом будут определены зависимые пакеты, и вам придется подтвердить, что вы согласны инсталлировать их. Весь процесс достаточно прост.

Совет

Вы не знаете, что такое `sshfs`? Зря! Прочитайте материал *Mount remote drives via SSH with SSHFS*, который доступен по адресу `http://opensource.weblogsinc.com/2005/11/03/mount-remote-drives-via-ssh-with-sshfs/`.

Удаление зависимых пакетов из системы Debian

```
apt -get remove [пакет]
```

Если пакет больше не нужен, команда `apt` позволяет без труда удалить его; для этого вместо `apt-get install` надо указать `apt-get remove`. Данная команда выполняет действия, противоположные `apt-get install`, — она удаляет указанные, а также зависимые пакеты. Как и ранее, при вызове команды надо задавать не имя файла, а имя пакета, например `apt-get remove sshfs`, а не `apt-get remove sshfs_1.1-1_i386.deb`.

```
# apt-get remove sshfs
Password:

The following packages will be REMOVED:
  sshfs

After unpacking 98.3kB disk space will be freed.
Do you want to continue [Y/n]?
```

Следует заметить, что процедура удаления пакета не затрагивает некоторые файлы, необходимые для его рабо-

ты. Так, например, в системе остаются конфигурационные файлы удаленных пакетов. Если вы уверены, что хотите удалить все элементы, вам надо задать опцию --purge.

```
# apt-get --purge remove sshfs
Password:

The following packages will be REMOVED:
  sshfs*

After unpacking 98.3kB disk space will be freed.
Do you want to continue [Y/n]?
```

Опцию --purge при удалении пакетов можно сравнить с символом звездочки при обычной операции с файлами. Удаляются все элементы пакета, включая конфигурационные файлы.

Обновление зависимых пакетов в системе Debian

```
apt -get upgrade
```

В современных системах Linux присутствуют тысячи пакетов, и в любой момент времени можно поручиться, что по крайней мере один из них нуждается в обновлении. Программа apt позволяет достаточно просто следить за программными пакетами и обновлять их. Процедура обновления выглядит следующим образом (напоминаю, что в системе K/Ubuntu, вместо того, чтобы запускать команду с полномочиями root, можно использовать sudo):

```
# apt-get update
Get:1 http://us.archive.ubuntu.com breezy
Release.gpg [189B]
Get:2 http://archive.ubuntu.com breezy Release.
```

```
gpg [189B]

Hit ftp://ftp.free.fr breezy/free Sources
Hit ftp://ftp.free.fr breezy/non-free Sources
Fetched 140kB in 1m4s (2176B/s)
Reading package lists... Done
[Результаты сокращены для экономии места]
# apt-get upgrade
Reading package lists... Done
Building dependency tree... Done
The following packages have been kept back:
  koffice
The following packages will be upgraded:
  kalzium kamera kanagram karbon kbruch kchart
kcoloredit kdegraphics kdegraphics-kfile-plugins
...
53 upgraded, 0 newly installed, 0 to remove and 1
not upgraded.
Need to get 58.3MB of archives.
After unpacking 28.7kB of additional disk space
will be used.
Do you want to continue [Y/n]?
```

Рассмотрим данный процесс более подробно. В данном случае мы, как и прежде, вызываем `apt-get update`, чтобы привести информацию на компьютере в соответствие со списком на сервере. Команда `apt-get upgrade` выявляет различия между инсталлированным программным обеспечением и данными в хранилище. Если такие различия имеют место, команда `apt` отображает список всех пакетов, которые надо скопировать и инсталлировать на компьютер. Реальное состояние списка пакетов зависит от того, как давно обновлялись программные средства. В данном примере с тех пор прошло достаточно много времени, и 53 пакета требуют обновления.

Если вы введете y, команда apt скопирует 53 пакета в каталог /var/cache/apt/archives и установит необходимые дополнения. Если вы не хотите обновлять программы, введите n.

Описанные действия достаточно просты, но программа apt позволяет еще эффективнее решить поставленную задачу. Вы можете объединить команды следующим образом:

```
# apt-get update && apt-get upgrade
```

Символы && указывают на то, что команда apt-get upgrade должна выполняться только в том случае, если команда apt-get update завершилась без ошибок. Можно также создать псевдоним в файле .bash_aliases (вопросы создания псевдонимов см. в главе 11).

```
alias upgrade='apt-get update && apt-get upgrade'
```

Перезагрузите файл .bash_aliases, и теперь, чтобы пакеты обновились, вам достаточно будет ввести upgrade, нажать клавишу <Enter>, а затем ввести y. Если сравнить данную процедуру с соответствующими средствами в системе Windows, оно будет явно не в пользу Windows Update.

Поиск пакетов, доступных для копирования в систему Debian

```
apt -cache search
```

Мы много говорили об инсталляции системы с помощью команды apt, но как узнать о доступных программных пакетах? Это также позволяет сделать команда apt, предоставляя инструмент apt-cache search, который ищет списки доступных пакетов в хранилище. Для вызова apt-cache search не обязательно иметь права root.

```
$ apt-cache search dvdcss
libdvdread3 - Simple foundation for reading DVDs
ogle - DVD player with support for DVD menus
libdvdcss2 - portable library for DVD decryption
libdvdcss2-dev - development files for libdvdcss2
```

Поиск, выполняемый данной командой, имеет одну особенность. Команда ищет совпадение последовательности символов, а не конкретных слов. Кроме того, шаблон поиска может присутствовать в имени пакета или описании. И наконец, `apt-cache search` просматривает весь список пакетов, как инсталлированных, так и удаленных, поэтому не исключено, что отображаемый пакет уже установлен у вас.

Совет

Существует продукт Synaptic, представляющий собой графический пользовательский интерфейс для apt. Он позволяет делать практически все, о чем шла речь выше, но вместо ввода данных вам достаточно щелкать мышью. В частности, очень удобны его средства поиска, поэтому лично я часто использую Synaptic исключительно для этой цели, а остальные действия выполняю из командной строки. Таким образом, удается использовать каждый инструмент максимально эффективно.

Удаление ненужных инсталляционных пакетов из системы Debian

```
apt -get clean
```

Пакеты, скопированные на локальный компьютер и инсталлированные на нем, система Debian сохраняет в каталоге /var/cache/apt/archives/. Со временем ненужные инсталляционные пакеты начинают занимать существенную часть дискового пространства. Удалить все не исполь-

зуемые файлы .deb позволяет команда apt-get clean (эта команда должна выполняться с правами root).

```
$ ls -1 /var/cache/apt/archives/
fuse-utils_2.3.0-1ubuntu1.1_i386.deb
libfuse2_2.3.0-1ubuntu1.1_i386.deb
lock
partial/
sshfs_1.1-1_i386.deb
# apt-get clean
$ ls -1 /var/cache/apt/archives/
lock
partial/
```

Если по каким-то причинам процесс копирования файлов будет прерван, скопированную часть пакета вы найдете в каталоге /var/cache/apt/archives/partial/. Если вы знаете, что все обновления и дополнения установлены, можете удалить содержимое данного каталога.

Устранение проблем с помощью команды apt

В процессе работы никто не застрахован от возникновения проблем. Некоторые из них описаны ниже, там же приводятся рекомендации, как справиться с ними посредством команды apt.

Предположим, что вы попытались выполнить apt-get, но получили следующее сообщение об ошибке:

```
E: Could not open lock file /var/lib/dpkg/lock -
open (13 Permission denied)
E: Unable to lock the administration directory
(/var/lib/dpkg/), are you root?
```

Причина проста: вы зарегистрированы не как пользователь root! Завершите сеанс работы, укажите при регистрации пользовательское имя root, и все будет работать.

На заметку

Если вы используете K/Ubuntu или другую систему, в которой вместо работы с правами root указывается sudo, то же сообщение об ошибке означает, что вы не задали sudo перед командой. Другими словами, команда, вызванная вами, имеет следующий вид:

```
$ apt-get upgrade
```

Чтобы не было ошибки, она должна выглядеть так:

```
$ sudo apt-get upgrade
```

Бывает, что команда apt сообщает о разрушенных зависимостях и предлагает выполнить команду `apt-get -f install`. Таким образом программа сообщает вам о том, что в системе есть разрушенные зависимости, не позволяющие завершить работу.

Существует несколько решений этой проблемы. Вы можете последовать совету apt и запустить команду `apt-get -f install`; она предпримет попытки разрешить ситуацию путем копирования и установки необходимых пакетов. Обычно этого бывает достаточно, чтобы привести все в порядок.

Если вы не хотите поступать подобным образом, попробуйте вызвать команду `apt-get -f remove`. Она удалит пакеты, которые, по мнению apt, являются источником некорректной работы. Такой шаг потенциально опасен, поэтому надо быть очень внимательным. Программа сообщает о предлагаемых изменениях и ожидает вашего подтверждения. Перед тем как ответить "yes", внимательно прочитайте сообщение apt.

И наконец, вы можете получить предупреждающее сообщение о том, что некоторые пакеты have been kept back. Этим команда apt информирует вас о том, что обнаружен конфликт между запрашиваемым пакетом (или одним из зависимых) и пакетом, уже инсталлированным в вашей

системе. Чтобы решить проблему, попробуйте инсталлировать пакет с опцией -u, которая сообщит вам о том, какие средства требуют обновления.

Выводы

Несмотря на то что системы на базе RPM и Debian имеют ряд различий, их средства управления пакетами все же во многом похожи. В обоих случаях разработчики позаботились об упрощении процедуры инсталляции и удаления программ. В RPM-системах для инсталляции, обновления и удаления программного обеспечения используется команда rpm, а в Debian той же цели служит команда dpkg. Существующие различия связаны с набором возможностей соответствующих инструментов и особенностями их использования. В частности, команда apt реально превосходит команду yum, но последняя постоянно дорабатывается.

В любом случае и apt, и yum значительно лучше, чем инструменты подобного назначения, доступные пользователям Windows. Windows Update обновляет только программное обеспечение Microsoft и несколько драйверов независимых производителей, в то время как apt и yum поддерживают практически любые программы, способные выполняться в системе Linux. Пользователи Linux имеют в своем распоряжении средства, лучшие, чем те, которые доступны пользователям Microsoft, и это заслуга разработчиков данной системы.

14

Сетевое взаимодействие

Средства сетевого взаимодействия присутствуют в системе Linux с момента ее появления. Они были включены в ядро небольшой группой программистов, и без них система, конечно же, не была бы столь популярной. Поддержка сетевого взаимодействия осуществляется большую часть времени, и соответствующие средства можно настраивать для выполнения самых разнообразных задач.

В этой главе вы узнаете о том, как проверять сетевые устройства и управлять ими. Когда все работает корректно (а так происходит почти всегда), вы можете использовать инструменты, описанные в данной главе, для контроля соединений. Если же вы столкнетесь с проблемой, они помогут вам решить ее, взяв на себя наиболее утомительную часть работы.

Совет

Материал этой главы основан на предположении о том, что вы используете систему адресации IPv4, следовательно, адрес представляется в формате `xxx.xxx.xxx.xxx`. В конце концов, IPv6 заменит IPv4, но это дело будущего. Тогда `route` и многие другие команды изменятся. На данный момент сведения, приведенные в настоящей главе, вполне могут быть использованы в работе. Дополнительная информация о IPv6 содержится в статье *IPv6* из Wikipedia (`http://en.wikipedia.org/wiki/Ipv6`).

Определение состояния сетевых интерфейсов

```
ifconfig
```

Данная глава полностью посвящена сетевым соединениям. В конце главы мы поговорим о том, как устранять проблемы, связанные с обменом по сети. Сейчас же выясним, как определить состояние сетевых средств.

Для того чтобы получить информацию о всех сетевых устройствах, независимо от того, работают они или нет, используйте команду `ifconfig` (она означает interface configuration) и задайте при ее вызове опцию `-a` (all). На экране отобразится информация наподобие следующей (заметьте, что в некоторых системах, для того, чтобы вызывать команду `ifconfig`, вам надо зарегистрироваться с правами root):

```
$ ifconfig -a
ath0 Link encap:Ethernet HWaddr 00:14:6C:06:6B:FD
inet addr:192.168.0.101 Bcast:192.168.0.255
Mask:255.255.255.0
inet6 addr: fe80::214:6cff:fe06:6bfd/64 Scope:Link
UP BROADCAST RUNNING MULTICAST MTU:1500 Metric:1
RX packets:1257 errors:7557 dropped:0 overruns:0
frame:7557
TX packets:549 errors:2 dropped:0 overruns:0
carrier:0
collisions:0 txqueuelen:200
RX bytes:195869 (191.2 KiB) TX bytes:95727
(93.4 KiB)
Interrupt:11 Memory:f8da0000-f8db0000

eth0 Link encap:Ethernet HWaddr 00:02:8A:36:48:8A
BROADCAST MULTICAST MTU:1500 Metric:1
```

```
RX packets:0 errors:0 dropped:0 overruns:0
frame:0
TX packets:0 errors:0 dropped:0 overruns:0
carrier:0
collisions:0 txqueuelen:1000
RX bytes:0 (0.0 b) TX bytes:0 (0.0 b)

lo Link encap:Local Loopback
inet addr:127.0.0.1 Mask:255.0.0.0
inet6 addr: ::1/128 Scope:Host
UP LOOPBACK RUNNING MTU:16436 Metric:1
RX packets:11092 errors:0 dropped:0 overruns:0
frame:0
TX packets:11092 errors:0 dropped:0 overruns:0
carrier:0
collisions:0 txqueuelen:0
RX bytes:982629 (959.5 KiB) TX bytes:982629
(959.5 KiB)
```

В данном примере приводится информация о трех интерфейсах: `ath0` (карта беспроводной связи), `eth0` (карта Ethernet) и `lo` (интерфейс обратной петли; он будет рассмотрен несколько позже). Для каждого из них, помимо всего прочего, указываются тип соединения, MAC-адрес (Media Access Control, или аппаратный адрес), IP-адрес, широковещательный адрес, маска подсети и информация о принятых и переданных пакетах. При разрыве соединения многие из этих данных теряются. Кстати, это один из способов определения состояния интерфейса. В данном примере вы видите, что `ath0` и `lo` работают, а `eth0` — нет, так как для него не указан IP-адрес и некоторые другие данные. Конечно, те же сведения можно получить проще: четвертая строка данных для `ath0` и `lo` начинается со слова `UP`, а для `eth0` это слово отсутствует.

Рассмотрим интерфейсы в порядке, обратном тому, в котором они представлены в листинге. Интерфейс `lo` называется интерфейсом обратной петли и позволяет компьютеру обращаться к самому себе. Интерфейс имеет IP-адрес 127.0.0.1 и необходим для нормальной работы системы. Если этот интерфейс присутствует, то в процессе работы вы не заметите этого, проблемы начнутся, если он по каким-то причинам исчезнет.

Совет

Дополнительную информацию об интерфейсе обратной петли и его адресе вы найдете в статье *Loopback* из Wikipedia (`http://en.wikipedia.org/wiki/Loopback`).

Интерфейс `eth0` — это карта Ethernet, к которой можно подключить кабель. В текущий момент кабель не подключен, поэтому интерфейс не активен и для него не отображаются IP-адрес, широковещательный адрес и маска подсети. Интерфейсы проводной и беспроводной связи могут работать одновременно, однако в большинстве случаев в этом нет необходимости.

И наконец, в системе присутствует `ath0` — PCMCIA-карта беспроводной связи. Имя `eth0` представляет первичный сетевой интерфейс, а `eth1` — вторичный. Когда беспроводная карта присутствует на компьютере, K/Ubuntu автоматически распознает ее и конфигурирует систему для работы с ней. Поскольку беспроводной интерфейс по сути представляет собой Ethernet-систему, информация о `ath0` и `eth0`, полученная посредством `ifconfig`, имеет в основном одинаковый формат.

На заметку

Для сетевых устройств могут использоваться другие имена, в частности карта беспроводной связи может иметь имя `wlan0`.

Команда `ifconfig -a` отображает информацию о всех интерфейсах, даже о неактивных. Если же опции при вызове команды `ifconfig` не указаны, приводятся сведения только об активных интерфейсах. Данная команда удобна в тех случаях, когда надо быстро выяснить состояние интерфейсов, в частности, если необходимо узнать их IP-адреса.

На заметку

Команда `ifconfig` позволяет также сконфигурировать сетевой интерфейс. Этот вопрос будет обсуждаться далее в данной главе.

Проверка способности компьютера принимать запросы

```
ping
ping -c
```

Команда `ping` передает на указанный адрес пакет специального типа — сообщение ICMP `ECHO_REQUEST`. Если компьютер по этому адресу принимает ICMP-сообщения, он отвечает пакетом ICMP `ECHO_REPLY`. (Заметьте, что брандмауэр может блокировать ICMP-сообщения, в этом случае команда `ping` становится бесполезной, однако в большинстве случаев она работает без проблем.) Успешное выполнение команды `ping` означает, что между двумя компьютерами поддерживается сетевое соединение.

```
$ ping www.google.com
ping www.google.com
PING www.l.google.com (72.14.203.99) 56(84)
bytes of data.
64 bytes from 72.14.203.99: icmp_seq=1 ttl=245
time=17.1 ms
64 bytes from 72.14.203.99: icmp_seq=2 ttl=245
```

```
time=18.1 ms

[Данные сокращены для экономии места]
--- www.l.google.com ping statistics ---
6 packets transmitted, 5 received, 16% packet
loss, time 5051ms
rtt min/avg/max/mdev=16.939/17.560/18.136/0.460 ms
```

Выполнение команды ping не прекратится до тех пор, пока вы не нажмете комбинацию клавиш <Ctrl+C>. Если вы забудете о том, что вызывали ping, эта команда будет выполняться до завершения работы компьютера или до разрыва соединения. (Однажды я забыл о ней, и она работала 18 дней, передав за это время на мой сервер 1,4 миллиона пакетов!)

Если вы хотите ограничить время работы ping, можете задать число пакетов, которые должна передать данная программа. Для этого надо задать опцию -c и ввести число. После того как ping передаст указанное количество пакетов, она прекратит работу, отобразив перед этим сообщение о результатах своего выполнения.

```
$ ping -c 3 www.granneman.com
PING granneman.com (216.23.180.5) 56(84) bytes of
data.
64 bytes from 216.23.180.5: icmp_seq=1 ttl=44
time=65.4 ms
64 bytes from 216.23.180.5: icmp_seq=2 ttl=44
time=64.5 ms
64 bytes from 216.23.180.5: icmp_seq=3 ttl=44
time=65.7 ms
--- granneman.com ping statistics ---
3 packets transmitted, 3 received, 0% packet
loss, time 4006ms
rtt min/avg/max/mdev=64.515/65.248/65.700/0.510 ms
```

Команда `ping` — стандартное средство, позволяющее быстро определить наличие сетевого соединения. Указав опцию `-c`, вы застрахуете себя от неожиданностей и через несколько дней не обнаружите, что она все еще работает.

Использование команды `ping` для диагностирования сетевых соединений будет рассмотрено далее в этой главе.

Контроль прохождения пакета между двумя узлами

```
traceroute
```

Команда `traceroute` отображает сведения о каждом шаге на пути пакета от вашей машины к указанному узлу. Предположим, вы хотите знать, почему вам не удается связаться с узлом `www.granneman.com`. Еще вчера все было нормально, но сегодня каждая попытка загрузить Web-страницу оканчивается по тайм-ауту. В чем проблема?

```
$ traceroute www.granneman.com
traceroute to granneman.com (216.23.180.5),
30 hops max, 38 byte packets
1 192.168.0.1 (192.168.0.1) 1.245 ms 0.827 ms
0.839 ms
2 10.29.64.1 (10.29.64.1) 8.582 ms 19.930 ms
7.083 ms
3 24.217.2.165 (24.217.2.165) 10.152 ms 25.476 ms
36.617 ms
4 12.124.129.97 (12.124.129.97) 9.203 ms 8.003 ms
11.307 ms
5 12.122.82.241 (12.122.82.241) 52.901 ms
53.619 ms 51.215 ms
6 tbr2-p013501.sl9mo.ip.att.net (12.122.11.121)
51.625 ms 52.166 ms 50.156 ms
7 tbr2-cl21.la2ca.ip.att.net (12.122.10.14)
```

```
50.669 ms 54.049 ms 69.334 ms
8 gar1-p3100.lsnca.ip.att.net (12.123.199.229)
50.167 ms 48.703 ms 49.636 ms
9 * * *
10 border20.po2-bbnet2.lax.pnap.net
(216.52.255.101) 59.414 ms 62.148 ms 51.337 ms
11 intelenet-3.border20.lax.pnap.net
(216.52.253.234) 51.930 ms 53.054 ms 50.748 ms
12 v8.core2.irv.intelenet.net
(216.23.160.66) 50.611 ms 51.947 ms 60.694 ms
13 * * *
14 * * *
15 * * *
```

Что означают символы * * *? Каждое появление этой последовательности соответствует пятисекундному тайм-ауту для соответствующего узла. В некоторых случаях это может означать, что машина вследствие сбоя попросту не может понять, как обработать пакет, но постоянное повторение символов * * * указывает на то, что источником проблемы является один из маршрутизаторов, в данном случае тот, которому узел `v8.core2.irv.intelenet.net` должен передавать пакеты. В такой ситуации вам надо оповестить администратора `v8.core2.irv.intelenet.net` о происходящем (не помешает также послать администратору `gar1-p3100.lsnca.ip.att.net` сообщение о том, что при попытке доступа к `border20.po2-bbnet2.lax.pnap.net` возникали проблемы, но в первую очередь, конечно, надо разобраться с узлом, находящимся за `v8.core2.irv.intelenet.net` по пути следования пакета).

Некоторые проблемы можно разрешить, увеличив число переходов, отслеживаемых `traceroute`. По умолчанию их количество равно 30, но вы можете увеличить это значение с помощью опции `-m`, например, задав команду `traceroute -m 40 www.bbc.co.uk`.

Совет

Самым лучшим инструментом для отслеживания прохождения пакетов, наверное, является `mtr` (имя этой программы расшифровывается как Matt's traceroute). Его можно представить себе как сочетание `ping` и `traceroute`. Если `mtr` может выполняться в вашей версии Linux, скопируйте эту программу и попробуйте поработать с ней. Дополнительную информацию вы найдете по адресу `http://www.bitwizard.nl/mtr/`.

Выполнение DNS-преобразования

```
host
```

Система доменных имен (Domain Name System — DNS) предназначена для того, чтобы упростить доступ пользователей к ресурсам глобальной сети. Компьютер лучше всего работает с числами (это неудивительно, ведь содержимое его памяти представляет собой именно числа), однако людям гораздо проще оперировать со словами. Web-узел может находиться по адресу 72.14.203.99, но большинству пользователей запомнить его будет очень трудно. Гораздо проще удержать в памяти имя `www.google.com`. DNS, по сути, представляет собой огромную базу данных, которая содержит информацию о соответствии миллионов IP-адресов доменным именам, в частности, о том, что 72.14.203.99 — это тот же узел, что и `www.google.com`.

Совет

DNS — сложная, но в то же время прекрасно организованная система. Общие сведения о ней приведены в статье Domain Name System из Wikipedia по адресу `http://en.wikipedia.org/wiki/Dns`, а более подробные в работе Пола Албица (Paul Albitz) и Крикета Лью (Cricket Liu) *DNS and BIND*.

Для того чтобы быстро найти IP-адрес, соответствующий доменному имени, надо использовать команду host.

```
$ host www.granneman.com
www.granneman.com is an alias for granneman.com.
granneman.com has address 216.23.180.5
www.granneman.com is an alias for granneman.com.
www.granneman.com is an alias for granneman.com.
granneman.com mail is handled by 30
bhoth.pair.com.
```

Поскольку система DNS выполняет различные варианты поиска, ответ в данном случае состоит из пяти строк. Обнаружить интересующую нас информацию нетрудно: имя www.granneman.com соответствует адресу 216.23.180.5.

Можно также решить обратную задачу — найти доменное имя по IP-адресу.

```
$ host 65.214.39.152
152.39.214.65.in-addr.arpa domain name pointer
web.bloglines.com.
```

На заметку

Определить IP-адрес позволяют многие другие команды, но вызов host — самый эффективный способ решения этой задачи. Кроме того, host позволяет выполнить обратное преобразование, что не всегда возможно с помощью других команд.

В конце данной главы будут рассмотрены примеры того, как команда host может помочь в разрешении проблем, связанных с обменом по сети.

Настройка сетевого интерфейса

`ifconfig`

В первом разделе данной главы мы рассматривали использование команды `ifconfig` для получения информации о состоянии сетевых интерфейсов. Однако команда `ifconfig` предоставляет более широкие возможности — например, позволяет настраивать сетевые интерфейсы.

На заметку

С ее помощью вы можете выполнять с интерфейсом самые разнообразные действия, но здесь мы рассмотрим лишь несколько из них (более подробную информацию вы получите по команде `man ifconfig`).

Для того чтобы изменить IP-адрес Ethernet-карты, соответствующей интерфейсу `eth0`, на `192.168.0.125`, выполните приведенную ниже команду (практически все действия по команде `ifconfig` требуют полномочий `root`).

```
# ifconfig eth0 192.168.0.125
```

Для того чтобы запустить некоторые из инструментов сбора информации, передаваемой по сети, например печально известный Ethereal (именно этот инструмент чаще всего используют злоумышленники для перехвата передаваемых пакетов и извлечения из них секретных данных), необходимо сначала отключить режим фильтрации данных в сетевой карте. По умолчанию карта, соответствующая интерфейсу `eth0`, принимает только те пакеты, которые были направлены именно ей, но чтобы организовать сбор всей информации, передаваемой по сети, надо сообщить карте, чтобы она принимала и остальные пакеты.

```
# ifconfig eth0 promisc
```

После того как вы выполните эту команду, ваша карта будет принимать любой пакет, который попал в ее поле зрения. Обратите внимание на ключевое слово PROMISC в четвертой строке.

```
# ifconfig eth0
eth0 Link encap:Ethernet HWaddr 00:02:8A:36:48:8A
 inet addr:192.168.0.143 Bcast:192.168.0.255
Mask:255.255.255.0
 inet6 addr: fe80::202:8aff:fe36:488a/64
Scope:Link
 UP BROADCAST PROMISC MULTICAST MTU:1500
Metric:1
[Данные сокращены для экономии места]
```

После выполнения интересующих вас действий отключите режим сбора всей информации, т.е. включите фильтрацию.

```
# ifconfig eth0 -promisc
# ifconfig eth0
eth0 Link encap:Ethernet HWaddr 00:02:8A:36:48:8A
inet addr:192.168.0.143 Bcast:192.168.0.255
Mask:255.255.255.0
inet6 addr: fe80::202:8aff:fe36:488a/64
Scope:Link
UP BROADCAST MULTICAST MTU:1500 Metric:1
```

При желании вы можете даже изменить аппаратный MAC-адрес вашего сетевого устройства. Это бывает необходимо при попытках связать сетевую службу с конкретной машиной. Заменяя MAC-адреса, будьте внимательны: ошибка может привести к конфликтам с другими сетевыми устройствами. Если вы хотите заменить адрес, сперва с помощью той же команды `ifconfig` выясните MAC-адрес, используемый по умолчанию, чтобы вы могли восста-

новить его впоследствии (заметьте, что адрес в приведенной ниже команде недопустим, так что не старайтесь установить его).

```
# ifconfig eth0 hw ether 00:14:CC:00:1A:00
```

Команда `ifconfig` — это "краеугольный камень" сетевых интерфейсов. Внимательно разберитесь в работе этой команды и попытайтесь в полной мере использовать ее возможности.

Получение информации о состоянии сетевого интерфейса беспроводной связи

```
iwconfig
```

Команда `ifconfig` отображает состояние всех сетевых интерфейсов, в том числе беспроводной связи. Однако все данные, относящихся к интерфейсам беспроводной связи, получить не удается, поскольку `ifconfig` попросту не имеет сведений о них. Для получения максимально возможных сведений о картах, предназначенных для установления беспроводных соединений, надо вместо `ifconfig` использовать `iwconfig`.

```
$ iwconfig
lo no wireless extensions.
eth0 no wireless extensions.
ath0 IEEE 802.11g ESSID:"einstein"
Mode:Managed Frequency:2.437 GHz Access Point:
00:12:17:31:4F:C6
 Bit Rate:48 Mb/s Tx-Power:18 dBm Sensitivity=0/3
 Retry:off RTS thr:off Fragment thr:off
 Power Management:off
 Link Quality=41/94 Signal level=-54 dBm
Noise level=-95 dBm
 Rx invalid nwid:1047 Rx invalid crypt:0
```

```
Rx invalid frag:0
  Tx excessive retries:73 Invalid misc:73
Missed beacon:21
```

Теперь вы видите специальную информацию, предоставляемую iwconfig, — данные, относящиеся исключительно к беспроводной связи, например, тип карты (в данном случае 802.11g), ESSID, или сетевое имя (einstein), тип сети, к которой подсоединен компьютер, MAC-адрес точки доступа (00:12:17:31:4F:C6) и различные сведения о качестве соединения.

Команды ifconfig и iwconfig совместно сообщают пользователю всю необходимую информацию о сетевом интерфейсе беспроводной связи. Для настройки карт такого типа можно использовать либо команду ifconfig, либо, как вы увидите в следующем разделе, команду iwconfig.

Настройка сетевого интерфейса беспроводной связи

```
iwconfig
```

В предыдущем разделе мы использовали команду iwconfig для получения важной информации о карте беспроводной связи и о соединении. Команду iwconfig также можно использовать для настройки карты. Название этой команды не только созвучно ifconfig, но похожи также и выполняемые ими действия.

На заметку

С помощью команды iwconfig вы можете выполнять с интерфейсом самые разнообразные действия, но здесь мы рассмотрим лишь некоторые из них (более подробную информацию вы получите по команде man iwconfig).

Давно известны стандартные топологии сетей, в которых сигнал передается по кабелю. Это, например, шина, звезда и кольцо. С появлением беспроводных сетей появились новые структуры, в частности, приведенные ниже.

- Managed (точка доступа формирует сеть, к которой могут подключаться беспроводные устройства; эта топология используется в большинстве беспроводных сетей).
- Ad-Hoc (два или несколько устройств беспроводной связи, способных взаимодействовать друг с другом, формируют сеть).
- Master (устройство беспроводной связи, выполняющее функции точки доступа).
- Repeater (устройство беспроводной связи, которое перенаправляет пакеты другим устройствам).

Существуют и другие топологии, но перечисленные выше применяются наиболее часто. С помощью команды `iwconfig` можно указать карте, как действовать в соответствии с той или иной топологией сети.

Совет

Дополнительную информацию о звезде, шине, кольце и других топологиях сетей можно найти в статье *Network topology* из Wikipedia по адресу `http://en.wikipedia.org/wiki/Network_topology`.

```
# iwconfig ath0 mode ad-hoc
```

После имени интерфейса надо указать опцию `mode`, а за ней режим, который вы собираетесь использовать.

На заметку

Заметьте, что карта, с которой мы работаем в этом разделе, соответствует интерфейсу ath0; на вашем компьютере могут присутствовать `eth1`, `wlan0` и другие интерфейсы. Для того чтобы определить имена интерфейсов, вызовите команду `iwconfig`, не указывая параметров, так, как это делалось в примере из предыдущего раздела.

ESSID (Extended Service Set Identifier) — это имя беспроводной сети, к которой подключен ваш компьютер или к которой вы собираетесь подключить его. В большинстве случаев можно использовать имя ESSID any; это означает, что вы согласны выполнить требования сети, касающиеся, например, кодирования. Однако некоторые сети требуют указания конкретного ESSID.

```
# iwconfig ath0 essid lincoln
```

В данном случае мы подключились к беспроводной сети с ESSID `lincoln`. Как видите, для этого достаточно в составе команды установить опцию `essid`, а за ней ввести требуемое имя.

В настоящее время в сетях все чаще используется шифрование. Делается это для того, чтобы защитить пользователей от злоумышленников, перехватывающих сетевой трафик и выискивающих в нем конфиденциальные данные. Самым простым методом кодирования информации в беспроводных сетях является WEP (Wired Equivalent Privacy). Однако этот метод обеспечивает очень низкую степень защиты. Квалифицированный злоумышленник может расшифровать данные, поэтому в последнее время все чаще применяется WPA (Wi-Fi Protected Access). К сожалению, использовать WPA для работы с картами беспроводной связи в системе Linux чрезвычайно сложно, и этот вопрос не будет рассматриваться в данной книге. На сегодняшний день WEP применяется чаще всего, и, несмотря на

недостатки, он все же лучше, чем отсутствие кодирования. Тем не менее, используя данный метод, не надейтесь обеспечить реальную безопасность своих данных.

Совет

Дополнительную информацию о WEP и WPA можно получить в статьях из Wikipedia Wired Equivalent Privacy (http://en.wikipedia.org/wiki/Wi-Fi_Protected_Access) и Wi-Fi Protected Access (http://en.wikipedia.org/wiki/Wi-Fi_Protected_Access). Сведения о том, как обеспечить работу WPA в конкретных версиях Linux, приводятся в документе *Linux WPA/WPA2/IEEE 802.1X Supplicant* (http://hostap.epitest.fi/wpa_supplicant/). Если вы используете Windows-драйверы посредством ndiswrapper, вам желательно ознакомиться с документом *How to use WPA with ndiswrapper* (http://ndiswrapper.sourceforge.net/mediawiki/index.php/WPA).

WEP использует разделяемый ключ кодирования — пароль, который известен как в точке доступа, так и на вашей машине. Пароль может задаваться в двух форматах: в виде шестнадцатеричного числа или текстовой строки. Использовать можно любой формат, команда iwconfig понимает их оба. Если вы собираетесь задать шестнадцатеричное число, введите его после опции enc.

```
# iwconfig ath0 enc 646c64586278742a6229742f4c
```

Эта же опция задается и при использовании строки текста, но сама строка предваряется символами s:.

```
# iwconfig ath0 enc s:dldXbxt*b)t/L
```

Совет

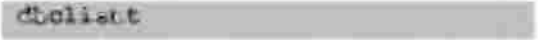
Приведенные здесь ключи WEP были сформированы с помощью очень удобного инструмента, который называется WEP Key Generator. Он доступен по адресу http://www.andrewscompanies.com/tools/wep.asp.

Если вам надо изменить сразу несколько характеристик беспроводного соединения, целесообразно сделать это с помощью одной команды. В этом случае после `iwconfig` надо указать имя устройства, а затем ввести информацию о всех изменениях, которые вы хотите выполнить.

```
# iwconfig ath0 essid lincoln enc 646c64586278742a6
229742f4c
```

В данном примере для устройства `ath0` указывается ESSID и задается WEP-шифрование посредством шестнадцатеричных цифр. Одна команда позволяет изменить любое число характеристик интерфейса.

Получение адресов средствами DHCP



В большинстве внутренних сетей организаций для выделения вновь подключаемым компьютерам IP-адресов и предоставления им другой информации о сети используются средства DHCP (Dynamic Host Control Protocol). Без DHCP соответствующие данные должны были бы быть жестко закодированы. При наличии DHCP новую машину достаточно присоединить к сети; DHCP-серверу будет передан запрос на предоставление IP-адресов, а ответ сервера будет автоматически учтен при формировании конфигурации сети.

На заметку

При изложении дальнейшего материала данной главы предполагается, что ваше сетевое устройство уже сконфигурировано для использования DHCP. В различных версиях Linux эта информация хранится в разных конфигурационных файлах. В системе Debian в файле `/etc/network/interfaces` содержится строка `iface` *интерфейс* `inet dhcp`. В Red Hat для этой цели в файл `/etc/sysconfig/network-`

scripts/ifcfg-*interface* включается выражение BOOTPROTO=dhcp. Дополнительную информацию по этому вопросу можно найти, выполнив средствами Google поиск по ключевым словам dhcp *версия_Linux*.

В некоторых случаях при загрузке машина не может обратиться к DHCP-серверу, и DHCP-запрос приходится инициировать вручную. Иногда возникают проблемы, для разрешения которых нужен новый IP-адрес. Независимо от того, для чего он потребовался, следует обратиться к одному из доступных DHCP-серверов посредством команды dhclient (она должна быть запущена от имени пользователя root).

```
# dhclient eth0
Listening on LPF/eth0/00:0b:cd:3b:20:e2
Sending on LPF/eth0/00:0b:cd:3b:20:e2
Sending on Socket/fallback
DHCPDISCOVER on eth0 to 255.255.255.255 port 67
interval 8
DHCPOFFER from 192.168.0.1
DHCPREQUEST on eth0 to 255.255.255.255 port 67
DHCPACK from 192.168.0.1
bound to 192.168.0.104 -- renewal in 37250
seconds.
# ifconfig eth0
eth0 Link encap:Ethernet HWaddr 00:0B:CD:3B:20:E2
inet addr:192.168.0.104 Bcast:192.168.0.255
Mask:255.255.255.0
inet6 addr: fe80::20b:cdff:fe3b:20e2/64
Scope:Link
```

Для того чтобы освободить выделенный вам адрес, надо указать при вызове команды dhclient опцию -r (release).

```
# dhclient -r eth0
sit0: unknown hardware address type 776
sit0: unknown hardware address type 776
Listening on LPF/eth0/00:0b:cd:3b:20:e2
Sending on LPF/eth0/00:0b:cd:3b:20:e2
Sending on Socket/fallback
```

В идеале команда `dhclient` должна автоматически выполняться при загрузке компьютера, установке новой PCMCIA-карты беспроводной связи или присоединении к обычной Ethernet-карте сетевого кабеля. Однако в некоторых случаях этого не происходит. Если средства DHCP работают не так, как положено, приходится явным образом вызывать `dhclient`. В процессе выполнения данная программа отображает подробную информацию о своих действиях; она помогает понять происходящее и выявить причины проблемы.

На заметку

В некоторых версиях Linux вместо `dhclient` для работы с DHCP используется более старая программа `pump`. Для того чтобы получить информацию о ней, выполните команду `man pump` либо обратитесь к документу *HOWTO for Red Hat and Mandrake*, доступному по адресу `http://www.faqs.org/docs/Linux-mini/DHCP.html#REDHAT6`.

Активизация сетевого соединения

```
ifup
```

В процессе работы вы систематически используете команду `ifup`, даже не зная об этом. Если ваш компьютер успешно подключился к Интернету, то это произошло благодаря `ifup`. Если вы подключили Ethernet-кабель к разъему на карте и через несколько секунд получаете возможность получать электронную почту, это значит, что команда `ifup`

выполнила большой объем работы. Реально ifup вызывается при возникновении события, имеющего отношение к сетевому взаимодействию, например перезагрузке компьютера, подключении кабеля и т.д. После запуска эта команда выполняет инструкции, описанные в конфигурационном файле (как вы помните, в предыдущем разделе упоминались имена и расположение этих файлов).

Иногда при возникновении проблем, связанных с обменом по сети, приходится вызывать команду ifup вручную. Сделать это просто: зарегистрируйтесь в системе как пользователь root, затем введите в командной строке ifup и имя сетевого интерфейса, который вы хотите активизировать.

```
# ifconfig
lo Link encap:Local Loopback
  inet addr:127.0.0.1 Mask:255.0.0.0
...
# ifup eth0
# ifconfig
eth0 Link encap:Ethernet HWaddr
00:0B:CD:3B:20:E2
inet addr:192.168.0.14 Bcast:192.168.0.255
Mask:255.255.255.0
inet6 addr: fe80::20b:cdff:fe3b:20e2/64 Scope:Link
UP BROADCAST RUNNING MULTICAST MTU:1500 Metric:1
...

lo Link encap:Local Loopback
inet addr:127.0.0.1 Mask:255.0.0.0
```

Команда ifup не сообщает об успешном завершении. Как и все Unix-программы, если все идет как надо, ifup "сохраняет молчание" и информирует пользователя лишь при возникновении ошибки. Для того чтобы получить

сведения о действиях, выполненных ifup, надо использовать команду ifconfig, как это было сделано в предыдущем примере.

На заметку

Для активизации сетевого соединения по кабелю или беспроводного соединения можно также использовать команду ifconfig *интерфейс* up или iwconfig *интерфейс* up.

Перевод сетевого интерфейса в неактивизированное состояние

```
ifdown
```

Если команда ifup активизирует сетевой интерфейс, то ifdown деактивизирует его. Зачем это может понадобиться? Предположим, например, что вы хотите активизировать интерфейс, а команда ifconfig сообщает вам, что он уже активизирован. Тогда надо перевести его в неактивизированное состояние, а затем снова активизировать.

```
# ifup eth0
ifup: interface eth0 already configured
# ifdown eth0
# ifup eth0
```

Как и ifup, команда ifdown в случае успешного выполнения не выводит информацию. Если вы не увидите никакого сообщения, это значит, что команда ifdown выполнила свою задачу и интерфейс деактивизирован.

На заметку

Для деактивизации сетевого интерфейса кабельной или беспроводной связи можно также использовать команды ifconfig *интерфейс* down и iwconfig *интерфейс* down.

Отображение таблицы маршрутизации

```
route
```

Пытаясь установить SSH-соединение с другим узлом (к этому вопросу мы еще вернемся в главе 15), необходимо как-то сообщить компьютеру, должен ли он пересылать пакеты только по локальной сети или обращаться к маршрутизатору, чтобы они были переданы в глобальную сеть. Аналогично, если ваш браузер обращается по адресу www.ubuntu.com, система должна знать, надо ли передавать запрос маршрутизатору

Решить эту задачу позволяет таблица маршрутизации, которая содержится в ядре Linux. Для того чтобы просмотреть текущее содержимое этой таблицы, надо ввести в командной строке команду `route` (просмотреть данные, содержащиеся в таблице маршрутизации, может любой пользователь, но чтобы изменить их, нужны права `root`).

```
$ route
Kernel IP routing table
Destination  Gateway    Genmask        Flags
  Metric Ref Use Iface
192.168.0.0  *          255.255.255.0  U
  0      0   0   eth0
default 1    92.168.0.1 0.0.0.0        UG
  0      0   0   eth0
```

На заметку

На данном компьютере имеется только один сетевой интерфейс, поэтому таблица маршрутизации чрезвычайно проста. Если бы на нем были установлены и Ethernet-карта, и устройство беспроводной связи, вы бы увидели дополнительные записи.

IP-адрес состоит из четырех октетов и записывается в формате xxx.xxx.xxx.xxx, например 192.168.0.124. Когда вы передаете пакет на другую машину, целевой IP-адрес сравнивается с данными в столбце Destination таблицы маршрутизации.

Совместно со столбцом Destination используется столбец Genmask, который позволяет определить, какие из четырех октетов в целевом адресе следует учитывать.

Предположим, например, что вы ввели в командной строке команду ping 192.168.0.124. Маска в столбце Genmask имеет вид 255.255.255.0, а это значит, что имеет значение лишь последний октет; он представлен числом 0.

Другими словами, в адресе 192.168.0.124 для выполнения маршрутизации важно только число 124. Пакеты, имеющие адреса от 192.168.0.1 до 192.168.0.255, соответствуют данным в столбцах Genmask и Destination, поэтому они должны пересылаться в пределах локальной сети. Именно поэтому в столбце Gateway указано значение *. При передаче по локальной сети шлюз не нужен.

Все остальные пакеты по умолчанию должны передаваться маршрутизатору, который имеет адрес 192.168.0.1; этот адрес указан в столбце Gateway. В этом случае маска, указанная в столбце Genmask, имеет значение 0.0.0.0. Другими словами, каждый пакет, для которого целевой адрес лежит вне диапазона 192.168.0.1–192.168.0.255, должен быть передан через компьютер с адресом 192.168.0.1 (он выполняет роль шлюза). Например, адреса 72.14.203.99, 82.211.81.166 и 216.23.180.5 соответствуют маске 0.0.0.0, поэтому они должны передаваться в глобальную сеть через компьютер, адрес которого указан в столбце Gateway.

Среди данных, отображаемых по команде route, есть столбец Flags. Он предоставляет дополнительную информацию о маршруте. Возможно несколько значений флагов, но чаще всего встречаются U (маршрут активизирован) и G

(использовать шлюз). В приведенном выше примере вы видите, что оба маршрута активизированы, но лишь второй из них предполагает передачу данных через шлюз.

Внесение изменений в таблицу маршрутизации

```
route
```

Команда `route` может использоваться не только для просмотра данных, содержащихся в таблице маршрутизации, но и для их изменения. Модифицируя таблицу, надо соблюдать осторожность, так как этим вы можете нарушить работу сети.

Предположим, например, что ваша машина не обращается к шлюзу, в результате чего пакеты не покидают локальную сеть (однажды это произошло на моей машине). Чтобы устранить неисправность, вам надо выполнить команду `route`, выяснить, что в столбце `Gateway` нет адреса маршрутизатора, а затем изменить данные с помощью той же команды `route` (несмотря на то, что команда `route` в режиме просмотра информации доступна любому пользователю, модифицировать таблицу маршрутизации может только пользователь `root`).

```
# route
Kernel IP routing table
Destination  Gateway  Genmask       Flags Metric
  MRef Use Iface
192.168.0.0  *        255.255.255.0 U     0
  0   0   eth0
# route add -net default gw 192.168.0.1 dev eth0
# route
Kernel IP routing table
Destination  Gateway      Genmask       Flags
  Metric Ref Use Iface
```

```
192.168.0.0     *               255.255.255.0 U
  0      0     0   eth0
default         192.168.0.1     0.0.0.0       UG
  0      0     0   eth0
```

Рассмотрим команду, приведенную в листинге. В ее составе имеется ключевое слово add, которое задает новый маршрут (удалить маршрут позволяет ключевое слово del). Опция -net сообщает ядру об адресе целевой сети; в данном случае имеется в виду целевой адрес, заданный по умолчанию, который определяется посредством ключевого слова default. Ключевое слово gw указывает на то, что пакеты, соответствующие данному пункту назначения, должны передаваться через шлюз с адресом 129.168.0.1 (поскольку здесь речь идет о маршруте по умолчанию, то значение маски равно 0.0.0.0). И наконец, выражение dev eth0 определяет используемое устройство, в данном случае это Ethernet-карта с интерфейсом eth0.

Предположим, что помимо Ethernet-карты в вашей машине появилась также карта беспроводной связи, соответствующая интерфейсу ath0. Вы хотите организовать через нее доступ к локальной сети 10.1.xxx.xxx, но не собираетесь передавать через нее информацию в глобальную сеть. Для того чтобы добавить соответствующий маршрут, надо выполнить команду, приведенную в следующем примере:

```
# route
Kernel IP routing table
Destination     Gateway         Genmask         Flags
  Metric Ref    Use Iface
192.168.0.0     *               255.255.255.0 U
  0      0     0   eth0
default         192.168.0.1     0.0.0.0       UG
  0      0     0   eth0
# route add -net 10.1.0.0 netmask 255.255.0.0
  dev ath0
```

```
# route
Kernel IP routing table
Destination  Gateway      Genmask        Flags
  Metric Ref Use Iface
192.168.0.0  *            255.255.255.0  U
  0      0   0   eth0
10.1.0.0     *            255.255.0.0    U
  0      0   0   ath0
default      192.168.0.1  0.0.0.0        UG
  0      0   0   eth0
```

Здесь мы указали, что карта соответствует устройству ath0, и задали маску 255.255.0.0. Теперь маршрутизация будет выполняться корректно. Если впоследствии вы захотите удалить этот маршрут, сделать это можно будет следующим образом:

```
# route
Kernel IP routing table
Destination Gateway      Genmask        Flags
  Metric Ref Use Iface
192.168.0.0 *            255.255.255.0  U
  0      0   0   eth0
10.1.0.0    *            255.255.0.0    U
  0      0   0   ath0
default     192.168.0.1  0.0.0.0        UG
  0      0   0   eth0
# route del -net 10.1.0.0 netmask 255.255.0.0
  dev eth0
# route
Kernel IP routing table
Destination Gateway      Genmask        Flags
  Metric Ref Use Iface
192.168.0.0 *            255.255.255.0  U
  0      0   0   eth0
```

```
default         192.168.0.1 0.0.0.0         UG
  0       0     0    eth0
```

Как видите, для этого используется та же команда, только вместо add вводится ключевое слово del.

Устранение проблем, связанных с сетевым взаимодействием

В настоящее время сетевые средства Linux разработаны столь тщательно, что пользователь вспоминает об их существовании только тогда, когда обмен по сети по каким-то причинам становится невозможным. Рассмотрим некоторые проблемы, которые могут возникнуть в процессе работы, и способы их устранения.

Если ваш сетевой интерфейс активизирован, но вы не можете взаимодействовать с другими узлами сети, попробуйте в первую очередь обратиться с помощью программы ping к собственному компьютеру, адрес которого 127.0.0.1. Если это не удается, значит, ваша система серьезно повреждена. Получив ответ с адреса 127.0.0.1, выполните следующий шаг: проверьте снова свой компьютер, но на этот раз установите в качестве параметра команды ping внешний адрес вашей машины. Если вы не получите ответа, убедитесь в том, что в вашей системе установлены сетевые средства. Если же отклик от компьютера получен, проверьте остальные машины в локальной сети. Отрицательный результат говорит о том, что что-то не в порядке с интерфейсом (предполагается, что маршрутизатор работает нормально). Убедитесь в том, что кабели подключены к соответствующим разъемам. Используйте ifconfig (или iwconfig в случае беспроводной связи) для проверки состояния интерфейсов, при необходимости активизируйте интерфейс с помощью команды ifup и снова попытайтесь выполнить команду ping.

Если компьютеры локальной сети откликаются на запросы ping, проверьте с помощью той же команды маршрутизатор. Если вы можете обмениваться данными с другими машинами в сети, но не можете взаимодействовать с маршрутизатором, значит, надо проверить с помощью команды route таблицу маршрутизации. Если в таблице нет нужных записей, добавьте их. Как это сделать, описано выше в данной главе.

На заметку

Диагностировать и устранять проблемы проще, если у вас есть образец конфигурационных файлов и других данных из работающей системы. Если вы знаете, что система функционирует правильно, выполните команду route и сохраните результаты. Вы сможете сверяться с ними, если таблица маршрутизации будет повреждена и вам придется восстанавливать ее.

Если маршрутизатор доступен, обратитесь с помощью программы ping к той машине из глобальной сети, о которой вы твердо знаете, что она в данный момент работает, например www.google.com или www.apple.com. Если вы не получите ответа, повторно вызовите команду ping, но на этот раз укажите IP-адрес компьютера. Конечно, это возможно только в том случае, если вы записали некоторые IP-адреса в записную книжку или в текстовый файл на компьютере. При необходимости вы можете использовать адреса, приведенные ниже. На данный момент они доступны, но не забывайте, что впоследствии они могут измениться.

Доменное имя	IP-адрес
www.google.com	72.14.203.99
www.apple.com	17.254.0.91
www.ubuntu.com	82.211.81.166
www.ibm.com	129.42.16.99
www.granneman.com	216.23.180.5

На заметку

Как были получены эти адреса? Определить их просто. Надо лишь вызвать команду `ping`, указав ей в качестве параметра доменное имя интересующего вас компьютера. В результате `ping` отобразит его IP-адрес. Ту же информацию можно получить с помощью команды `traceroute`. Еще быстрее можно выяснить IP-адрес с помощью команды `host`, которая рассматривалась ранее в этой главе.

Если обращение по IP-адресу возможно, а по доменному имени — нет, значит, источником проблемы является система DNS. Если вы используете DHCP, надо выполнить команду `dhclient`, чтобы обновить информацию о DNS, предоставляемую DHCP-сервером. Если DHCP не используется, проверьте установки в вашей системе, связанные с DNS (нужные данные можно получить у системного администратора или в службе поддержки провайдера), и при необходимости измените содержимое файла `/etc/resolv.conf`. В этом файле должна присутствовать информация, подобная приведенной ниже.

```
nameserver 24.217.0.5
nameserver 24.217.0.55
```

После ключевого слова `nameserver` следует IP-адрес DNS-сервера, который вы собираетесь использовать в работе. Если ваш маршрутизатор может выполнять функции DNS-сервера, то в первую очередь надо указать его адрес (например, 192.168.0.1). Тогда запись в файле `/etc/resolv.conf` будет иметь такой вид:

```
nameserver 192.168.0.1
```

Выполните команду `ifdown`, а затем `ifup` и проверьте, происходит ли обмен данными. Если проблема не устранена, надо начать все сначала, проверив в первую очередь аппаратные средства. Правильно ли выполнены установки?

Подключены ли кабели? Если вы уверены, что на эти вопросы можно дать положительные ответы, значит, пришло время проверить программное обеспечение. Хуже всего, если для используемых вами аппаратных средств в системе Linux нет драйверов. Это происходит редко, причем чем дальше, тем реже, но бывает, что все-таки случается.

Чаще всего несовместимыми с Linux бывают устройства беспроводной связи, так как некоторые производители не хотят взаимодействовать с разработчиками Linux. Для того чтобы предотвратить возникновение проблем, желательно, приобретая карту беспроводной связи, просмотреть материалы, опубликованные в глобальной сети, и выяснить, совместима ли она с вашей системой. Полезную информацию можно найти на серверах Hardware Supported by Madwifi (`http://madwifi.org/wiki/Compatibility`) и Linux wireless LAN support (`http://linux-wless.passys.nl/`). Данные, опубликованные на них, могут быть устаревшими, но все равно они окажут существенную помощь в работе. Постоянно обновляемый список ссылок, имеющих отношение к совместимости карт беспроводной связи с системой Linux, расположен по адресу `http://del.icio.us/rsgranne/wireless`. Если говорить о конкретных аппаратных средствах, то можно порекомендовать карту Netgear WG511T wireless PCMCIA. Лично я использую ее в своем компьютере, работающем под управлением последней версии K/Ubuntu, и с момента установки у меня не было претензий к ней.

Закончим разговор о проблемах. Если при вызове `ping` вы задаете IP-адрес или доменное имя и в обоих случаях получаете ответ, это означает, что взаимодействие по сети поддерживается!

Выводы

В данной главе мы рассмотрели самые разные инструменты для работы с сетевыми средствами. Многие из них, например `ifconfig`, `iwconfig` и `route`, могут решать разные задачи, а именно: информировать вас о состоянии средств обмена по сети и изменять их конфигурацию. Внекоторых случаях приходится прибегать к дополнительным средствам диагностирования проблем, например `ping`, `traceroute` и `host`. И наконец, ряд команд влияет на саму возможность взаимодействия по сети; это `dhclient`, `ifup` и `ifdown`. Если вы собираетесь серьезно работать в системе Linux, вам надо хорошо знать рассмотренные здесь инструменты. Отсутствие сетевого соединения очень неприятно, но во многих случаях правильно выбрав программу, вы сможете быстро устранить проблему.

15

Работа в сети

Многие из команд, рассматриваемых в данной главе, предоставляют столь обширные возможности, что для подробного обсуждения их потребовалась бы отдельная книга. Мы не будем даже пытаться выяснить все, что позволяют сделать `ssh`, `rsync`, `wget` и `curl`. Материал, изложенный в данной главе, можно рассматривать лишь как общие сведения о работе с этими инструментами. Самое лучшее, что вы можете сделать, — это попытаться использовать эти команды самостоятельно, освоить их основные функции, а затем постепенно расширять свои знания о них. Это может занять у вас достаточно много времени, но, освоив данные инструменты, вы наверняка придумаете самые различные применения для них.

Организация защищенного взаимодействия с другим компьютером

```
ssh
```

Поскольку разработчики Unix с самого начала позаботились о сетевом взаимодействии, неудивительно, что даже в самых ранних версиях системы имелись команды, позволяющие пользователям устанавливать соединения с другими машинами. Посредством этих соединений можно

было запускать программы, просматривать файлы и обращаться к ресурсам. В течение длительного времени основным средством для регистрации на удаленном компьютере была система Telnet, но она обладала существенным недостатком — в ней практически отсутствовала защита. Все данные, передаваемые с помощью Telnet, — имя пользователя, пароль, любые команды и данные — пересылались в незашифрованном виде. Любой, кто имел возможность перехватить сетевой трафик, мог получить в свое распоряжение всю передаваемую информацию.

Для того чтобы справиться с этой проблемой, была разработана система ssh (secure shell). С ее помощью можно было делать все, что позволяла система Telnet, и даже много больше. Кроме того, весь трафик был закодирован, что делало систему ssh намного более полезной для пользователей, чем Telnet. Если вам необходимо установить взаимодействие с другим компьютером, независимо от того, находится ли он на противоположном конце земного шара или в соседней комнате, используйте для этой цели ssh.

Предположим, что вы хотите обратиться средствами ssh с вашего портативного компьютера (он имеет имя pound и адрес 192.168.0.15) к настольной системе (с именем eliot и адресом 192.168.0.25) для просмотра файла. Пусть на портативном компьютере ваше пользовательское имя ezra, а на настольной системе для вас создана учетная запись tom. Чтобы организовать взаимодействие, вам надо выполнить следующую команду (если необходимо, при ее вызове можно также использовать доменное имя, например hoohah.granneman.com):

```
$ ssh tom@192.168.0.25
tom@192.168.0.25's password:
Linux eliot 2.6.12-10-386 #1 Mon Jan 16 17:18:08
UTC 2006 i686 GNU/Linux
```

```
Last login: Mon Feb 6 22:40:31 2006 from
192.168.0.15
[Данные сокращены для экономии места]
```

После установки соединения система запросит у вас пароль. Введите его (вы не увидите пароль на экране; это сделано для того, чтобы его не увидели и другие пользователи, которые могут в это время случайно оказаться рядом) и нажмите клавишу <Enter>. Если пароль задан правильно, вы увидите некоторую информацию о машине, к которой вы только что подключились, в частности, ее имя, версию системы и время предыдущей регистрации. Теперь вы можете вызывать на удаленной машине любые команды в пределах ваших полномочий, причем они буду выполняться так, как будто вы работаете за ее терминалом. С точки зрения `ssh` и `eliot` ваше расположение несущественно — вы зарегистрированы и можете работать в системе.

Если вы обращаетесь к `eliot` впервые, то можете увидеть следующее сообщение:

```
$ ssh tom@192.168.0.25
The authenticity of host '192.168.0.25
(192.168.0.25)' can't be established.
RSA key fingerprint is 54:53:c3:1c:9a:07:22:0c:82:
7b:38:53:21:23:ce:53.
Are you sure you want to continue connecting
(yes/no)?
```

Система `ssh` сообщает, что она не может распознать вашу машину, и предлагает подтвердить ее подлинность. Введите `y`, нажмите клавишу <Enter>, и вы получите другое сообщение с приглашением для ввода пароля.

```
Warning: Permanently added '192.168.0.25' (RSA)
to the list of known hosts.1
tom@192.168.0.25's password:
```

С этого момента взаимодействие будет происходить как обычно. Приведенное выше сообщение вы увидите лишь при первом соединении с `eliot`. В дальнейшем система `ssh` сохранит ключ RSA на компьютере pound в файле `~/.ssh/known_hosts`. Откройте этот файл с помощью текстового редактора, и вы увидите, что в нем появилась новая строка. В зависимости от состояния опции `HashKnownHosts` в файле `/etc/ssh/ssh_config` эта строка будет записана в одном из двух форматов. Если значение опции равно `no`, эта строка будет выглядеть следующим образом:

```
192.168.0.25 ssh-rsa SkxPUQLYqXSzknsstN6
Bh2MHK5AmC6Epg4psdNL69R5pHbQi3kRWNNNNO3
AmnP1lp2RNNNNOVjNN9mu5FZel6zKOiKfJBbLh/
Mh9KOhBNtrX6prfcxO9vBEAHYITeLTMmYZLQHBxSr6ehj/
9xFxkCHDYLdKFmxaffgA6Ou2ZUX5NzP6Rct4cfqAY69E5cUoDv3xEJ/
gj2zv0bh630zehrGc=
```

IP-адрес присутствует в ней в явном виде вместе с закодированными данными. Если же значение `HashKnownHosts` равно `yes`, то эта же строка будет иметь следующий вид:

```
NNNNO3AmnP1lp2RNNNNOVjNNNVRNgaJdxOt3GIrhOOlPD6
KBIU1kaT6nQoJUMVTx2tWb5KiF/LLD4Zwbv2Z/j/OczCZI
QNPwDUf6YiKUFFC6eagqpLDDB4T9qsOajOPLNinRZpcQoP
lXf1u6j1agfJzqUJUYE+Lwv8yzmPidCvOuCZOLQH4qfkVN
XEQxmyy6iz6b2wp=?
```

Теперь закодированы все данные, включая IP-адрес или доменное имя машины. С точки зрения безопасности такая запись предпочтительнее, однако если на компьютере `eliot` будет установлена другая операционная система, то возникнет проблема. Вновь обратившись после переустановки операционной системы к `eliot`, вы увидите следующее устрашающее сообщение:

```
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
@ WARNING: REMOTE HOST IDENTIFICATION HAS CHANGED! @
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
IT IS POSSIBLE THAT SOMEONE IS DOING SOMETHING NASTY
Someone could be eavesdropping on you right now
(man-in-the-middle attack)!
It is also possible that the RSA host key has just
been changed.
The fingerprint for the RSA key sent by the remote
host is
19:85:59:5c:6a:24:85:53:07:7a:dc:34:37:c6:72:1b.
Please contact your system administrator.
Add correct host key in /home/pound/.ssh/known_hosts
to get rid of this message.
Offending key in /home/pound/.ssh/known_hosts:8
RSA host key for 192.168.0.125 has changed and you
have requested strict checking.
Host key verification failed.
```

Проблема состоит в том, что ключ ssh на компьютере eliot при установке операционной системы изменился. Средства поддержки ssh обнаруживают различие между ключом, хранящимся в файле known_hosts на компьютере pound, и новым ключом и предупреждают вас об этом. Вам следует удалить из файла known_hosts строку, соответствующую eliot, сохранить файл и повторно установить соединение. Теперь с точки зрения ssh это будет ваше первое соединение, и вам будет предложено принять значение ключа. Согласитесь с этим предложением, и дальнейшее взаимодействие будет проходить, как и прежде.

Если eliot — единственная машина, с которой вы взаимодействуете посредством ssh, то найти нужную строку в файле known_hosts будет несложно, так она там окажется единственной. Если же вы устанавливаете соединение с не-

сколькими машинами, задача усложнится. Действительно, какая из приведенных ниже строк представляет компьютер hoohah.granneman.com?

```
AAAAB3NzaC1yc2EAAAABIwAAAIEAtnWqkBg3TVeu00yCQ6XOVH1
xnG6aDbWHZIGk2gJo5XvS/YYQ4Mjoi2M/w/0pmPMVDACjQHs6Lv
XHSSP6rntdcYQQO4G9dfBnwBCYAvaEMcpDbCyKs1h6w1ntsWmdH
WHLR+Yji8lmzCvqPiBhPM0YDU4dsxIAKRDkzll6vm6o2jc=

NNNNO3AmnP1lp2RNNNNOVjNNNVRNgaJdxOt3GIrhOOlPD6KBIU1
kaT6nQoJUMVTx2tWb5KiF/LLD4Zwbv2z/j/OczCZIQNPwDUf6Yi
KUFFC6eagqpLDDB4T9qsOajOPLNinRZpcQoPlXf1u6j1agfJzqU
JUYE+Lwv8yzmPidCvOuCZOLQH4qfkVNXEQxmyy6iz6b2wp=

AAAAB3NzaC1yc2EAAAABIwAAAIEAtnWqkBg3TVeu00yCQ6XOVH1
xnG6aDbWHZIGk2gJo5XvS/YYQ4Mjoi2M/w/0pmPMVDACjQHs6Lv
XHSSP6rntdcYQQO4G9dfBnwBCYAvaEMcpDbCyKs1h6w1ntsWmdH
WHLR+Yji8lmzCvqPiBhPM0YDU4dsxIAKRDkzll6vm6o2jc=
```

Поскольку вы не можете дать ответ на этот вопрос, не остается ничего, кроме как удалить все строки, а это значит, что вам придется принять ключ при первой регистрации на каждой машине. Для того чтобы предотвратить возникновение подобной ситуации, можно зарегистрироваться как пользователь root, открыть файл /etc/ssh/ssh_config и установить значение HashKnownHosts равным no.

Защищенная регистрация на другой машине без использования пароля

```
ssh
```

На первый взгляд может показаться, что в название данного раздела вкралась ошибка, однако это не так. Действительно, есть возможность зарегистрироваться средствами ssh, но не указывать при этом пароль. Если вы ежедневно работаете с одним и тем же компьютером (лично мне приходится регистрироваться на некоторых системах

несколько раз в день), то подход, описанный в этом разделе, может оказаться очень полезным.

Предположим, что вы хотите, чтобы при регистрации на компьютере `eliot` (пользовательское имя `tom`) с машины `pound` (пользовательское имя `ezra`) можно было не вводить пароль. Для этого прежде всего надо создать на компьютере `pound` аутентификационный ключ `ssh`, используя следующую команду:

```
$ ssh-keygen -t dsa
Generating public/private dsa key pair.
Enter file in which to save the key
(/home/ezra/.ssh/id_dsa):
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in
/home/ezra/.ssh/id_dsa.
Your public key has been saved in
/home/ezra/.ssh/id_dsa.pub.
The key fingerprint is:
30:a4:a7:31:27:d1:61:82:e7:66:ae:
ed:6b:96:3c:24ezra@pound
```

Примите расположение ключа по умолчанию, нажав клавишу <Enter>, и оставьте поле `passphrase` пустым, дважды нажав в ответ на соответствующее приглашение клавишу <Enter>. Таким образом вы создадите закрытый ключ в файле `~/.ssh/id_dsa` и открытый ключ в файле `~/.ssh/id_dsa.pub`.

Теперь вам надо передать открытый ключ (но не закрытый!) с компьютера `pound` на компьютер `eliot`. Разработчики системы `ssh` позаботились о вас и создали программу, которая делает это быстро и без ошибок. Для того чтобы автоматически скопировать открытый ключ с компьютера `pound` на компьютер `eliot`, введите следующую команду:

```
$ ssh-copy-id -i ~/.ssh/id_dsa.pub tom@192.168.0.25
```

Теперь попытайтесь зарегистрироваться, используя ssh-идентификатор tom@192.168.0.25, и убедитесь, что в файле .ssh/autorized_keys не появились дополнительные ключи.

Все необходимые действия выполнены (если вы хотите последовать дополнительным рекомендациям, отображаемым при выполнении ssh-copy-id, можете сделать это). Выясним, что произойдет, если вы теперь обратитесь с компьютера pound на компьютер eliot средствами ssh.

```
$ ssh tom@192.168.0.25
Linux eliot 2.6.12-10-386 #1 Mon Jan 16 17:18:08
UTC 2006 i686 GNU/Linux
Last login: Mon Feb 6 22:40:31 2006 from
192.168.0.15
```

Заметьте, что вам не предлагалось ввести пароль. Именно такого поведения системы вы и добивались.

Возможно, у вас возникнут сомнения в том, обеспечивает ли подобный подход реальную защиту, поскольку пароль отсутствует, а ключ передается свободно. Действительно, это так, но давайте подробнее разберемся в происходящем. Если кто-либо сможет проникнуть на компьютер pound, он также сможет без проблем соединиться с компьютером eliot, так как пароль не запрашивается. Но ведь вы приняли меры к защите системы pound. Если этот компьютер удастся взломать, вы будете иметь достаточно серьезные проблемы, независимо от того, поймет ли злоумышленник, что он может также проникнуть на eliot. С другой стороны, пароли постоянно передаются по сети. Если кто-то посторонний сумеет получить его, он получит доступ к компьютеру и сможет использовать хранящуюся на нем информацию в своих целях. Но ведь закрытый ключ так

же важен, как и пароль, и его также надо надежно хранить. Немного поразмыслив, вы придете к выводу, что обмен ключами посредством ssh обеспечивает не меньший, а во многих случаях более высокий уровень защиты, чем использование паролей.

Если вы все же предпочитаете пароли — вы вольны поступать так, как вам хочется. Одно из преимуществ открытых систем и, в частности, Linux — возможность свободного выбора.

Защищенная система FTP

```
sftp
```

Подобно тому, как ssh во многим превосходит Telnet, система SFTP предпочтительнее, чем обычные средства FTP. Как и Telnet, FTP передает по сети пользовательское имя и пароль, не кодируя их, и любой, кто имеет возможность перехватывать пакеты, сможет узнать их. В отличие от FTP, SFTP использует SSH для шифрования всех данных: пользовательского имени, пароля и остального трафика. Если не учитывать тот факт, что SFTP обеспечивает в миллион раз более надежную защиту, чем FTP, то в остальном эти системы очень похожи друг на друга. В них используются одинаковые системы команд, в результате пользователь, который привык работать с FTP, без труда перейдет на SFTP.

Если вы можете получить доступ к машине средствами SSH, то вы также можете работать с ней с помощью SFTP. Для того чтобы вызвать на компьютере pound (192.168.0.15; пользовательское имя ezra) команду sftp, которая позволит взаимодействовать с машиной eliot (192.168.0.25; пользовательское имя tom), надо ввести в командной строке следующее выражение:

```
$ sftp tom@192.168.0.25
Connecting to 192.168.0.25...
tom@192.168.0.25's password:
sftp>
```

Если вы внимательно прочитали предыдущий раздел, то, возможно, спросите, почему система предлагает ввести пароль? Вы правы, в данный момент этого не должно быть. Дело в том, что этот пример был получен еще до того, как система была настроена для установления соединения без пароля. Если же мы будем работать с системой `ssh`, не запрашивающей пароль, то процедура регистрации будет выглядеть следующим образом:

```
$ sftp tom@192.168.0.25
Connecting to 192.168.0.25...
sftp>
```

После регистрации в системе SFTP можно использовать стандартные команды для работы с каталогами и передачи файлов. Некоторые из наиболее часто применяемых команд описаны в табл. 15.1. Полный список операций можно получить, выполнив команду `man sftp`.

Таблица 15.1. Наиболее часто используемые команды SFTP

Команда	Описание
`cd`	Перейти в другой каталог
`exit`	Закрыть соединение с удаленным SSH-сервером
`get`	Скопировать указанный файл на локальную машину
`help`	Отобразить справочную информацию
`lcd`	Перейти в другой каталог на локальной машине
`lls`	Отобразить список файлов на локальной машине
`ls`	Отобразить список файлов в текущем каталоге на удаленном SSH-сервере

Окончание табл. 15.1

Команда	Описание
put	Скопировать указанный файл на удаленный SSH-сервер
rm	Удалить указанный файл с удаленного SSH-сервера

Защищенное копирование файлов между узлами сети

`scp`

Если у вас мало времени и вам надо скопировать файл с одной машины на другую, обеспечив при этом защиту, вам поможет команда `scp` (secure copy). Данная команда вызывается следующим образом:

```
scp пользователь@узел_1:файл_1 пользователь@узел_2:
файл_2
```

Как видите, синтаксис похож на известную вам команду `cp`, "расширенную" для работы в сети. Для того чтобы подробнее ознакомиться с работой данной команды, рассмотрим пример. Предположим, что вам надо скопировать с помощью `scp` файл `backup.sh` с компьютера `pound` (192.168.0.15; пользовательское имя `ezra`) в каталог `/home/tom/bin` на компьютере `eliot` (129.168.0.25; пользовательское имя `tom`).

```
$ pwd
/home/ezra
$ ls ~/bin
backup.sh
$ scp ~/bin/backup.sh tom@192.168.0.25/home/tom/bin
backup.sh 100% 8806 8.6KB/s 00:00
$
```

Пароль не запрашивается. Дело в том, что scp работает на основе системы ssh, которую мы в предыдущем разделе настроили для регистрации без указания пароля. Если в вашей системе такая настройка не была выполнена, то, перед тем, как продолжить работу, система предложит вам ввести пароль пользователя tom.

Допустим, что у вас есть несколько JPEG-файлов, которые вы хотите скопировать с компьютера pound на компьютер eliot. Сделать это очень просто, надо лишь применить символ групповой операции.

```
$ ls -1 ~/covers
earth_wind_&_fire.jpg
handel_-_chamber_music.jpg
smiths_best_1.jpg
strokes_-_is_this_it.jpg
u2_pop.jpg
$ scp *.jpg tom@192.168.0.25:/home/tom/album_covers
earth_wind_&_fire.jpg        100%   44KB   43.8KB/s
handel_-_chamber_music.jpg   100%   12KB   12.3KB/s
smiths_best_1.jpg            100%   47KB   47.5KB/s
strokes_-_is_this_it.jpg     100%   38KB   38.3KB/s
u2_pop.jpg                   100% 9222
9.0KB/sQ
```

Теперь предположим, что вам надо выполнить копирование в обратном направлении. Вы по-прежнему работаете с компьютером pound и хотите скопировать несколько изображений с eliot на компьютер pound в каталог, отличный от текущего.

```
$ scp tom@192.168.0.25:/home/tom/pictures/dog/
libby* ~/pix/libby
libby_in_window_1.20020611.jpg  100%  172KB
172.4KB/s
```

```
libby_in_window_2.20020611.jpg  100%  181KB
180.8KB/s
libby_in_window_3.20020611.jpg  100%  197KB
196.7KB/s
libby_in_window_4.20020611.jpg  100%  188KB
187.9KB/s
```

Как видите, команда `scp` очень удобна, если вам надо выполнить защищенное копирование файлов между различными машинами. Если же вам надо скопировать большое количество файлов, работа с данной командой вскоре станет для вас утомительной. Возможно, в этом случае будет более целесообразно воспользоваться системой SFTP или организовать обмен с диском, смонтированным средствами Samba (система Samba будет рассмотрена в следующей главе).

Защищенная передача файлов и создание резервных копий

```
rsync -v
```

Команда `rsync` — великолепный, чрезвычайно полезный инструмент, который многие пользователи (и я в их числе) применяют ежедневно. Что делает эта команда? Она может решать неисчислимое множество задач (это одна из тех команд, о которых можно написать отдельную книгу), но здесь мы сосредоточим внимание на ее самой необходимой функции — способности эффективно создавать резервные копии, затрачивая на это минимальный объем сетевого трафика.

Предположим, что вы хотите каждый вечер создавать копии файлов общим объемом 2 Гбайт, находящихся на машине `coleridge` (пользовательское имя `sam`) и хранить их на другом компьютере, `wordsworth` (пользовательское имя `will`). Если бы в вашем распоряжении не было команды

rsync, вам пришлось бы каждый вечер копировать 2 Гбайт информации. Это была бы значительная нагрузка даже на быстродействующие линии связи. Благодаря команде rsync необходимую информацию удается передать почти мгновенно. Почему так происходит? Дело в том, что при формировании резервных копий команда rsync передает лишь информацию о различиях между файлами. Если за последние 24 часа изменение претерпели только несколько сотен килобайт, то именно такой объем информации и будет передан. Если же модифицированы были 100 Мбайт, команда rsync передаст этот объем данных. В любом случае это намного меньше, чем 2 Гбайт.

Ниже приведена команда, которая запускается на компьютере coleridge и передает содержимое каталога с документами на диск компьютера wordsworth, предназначенный для хранения резервных копий. Ознакомьтесь с командой и результатами ее выполнения и приготовьтесь к рассмотрению конкретных опций. Как видите, в листинге приведены два варианта одной и той же команды: в первом использованы "длинные" имена опций, а во втором некоторые опции представлены в виде одной буквы:

```
$ rsync --verbose --progress --stats -recursive
  --times --perms --links --compress --rsh=ssh
  --delete /home/sam/documents/
  will@wordsworth:/media/backup/documents

$ rsync -v --progress --stats -r -t -p -l -z
  -e ssh --delete /home/sam/documents/
  will@wordsworth:/media/backup/documents
```

Ту же команду можно записать и так:

```
$ rsync -vrtplze ssh --progress --stats
  -delete /home/sam/documents/
  will@wordsworth:/media/backup/documents
```

Независимо от способа вызова, вы получите результаты наподобие приведенных ниже.

```
building file list ...
107805 files to consider
deleting clientele/Linux_Magazine/do_it_
yourself/13/gantt_chart.txt~
deleting Security/diebold_voting/black_box_voting/
bbv_chapter-9.pdf

deleting E-commerce/Books/20050811 eBay LIL ABNER
DAILIES 6 1940.txt
Security/electronic_voting/diebold/black_box_
voting/bbv_chapter-9.pdf
legal_issues/free_speech/Timeline A history of
free speech.txt
E-commerce/2005/Books/20050811 eBay LIL ABNER
DAILIES 6 1940.txt

connectivity/connectivity_info.txt
[Набор результатов существенно сокращен для
экономии места]
Number of files: 107805
Number of files transferred: 120
Total file size: 6702042249 bytes
Total transferred file size: 15337159 bytes

File list size: 2344115

Total bytes sent: 2345101
Total bytes received: 986
sent 2345101 bytes received 986 bytes 7507.48
bytes/sec
total size is 6702042249 speedup is 2856.69
```

Сначала rsync формирует список файлов, предназначенных для обработки (в данном случае их количество равно 107805), затем удаляет с целевого носителя (wordsworth) те файлы, которых больше не существует на исходном носителе (coleridge). В данном примере удалены три файла: резервная копия статьи из Linux Magazine, PDF-файл с результатами электронного голосования и один текстовый файл.

После удаления файлов команда rsync заменяет файлы, подвергшиеся изменениям, новыми версиями. В данном случае копируются четыре файла. Несмотря на то что два PDF-файла лишь были перемещены в другой подкаталог, для команды rsync они являются новыми файлами, поэтому программа копирует их. Файл A history of free speech.txt — новый, поэтому он также копируется на wordsworth.

После вывода сообщений об изменениях команда rsync предоставляет общие сведения о переданных данных. В данном случае скопированы были 120 файлов, 15337159 байтов (около 14 Мбайт) из 6702042249 (приблизительно 6,4 Гбайт). Отображаются и другие данные, но эти — основные.

Теперь поговорим о том, какие действия вы предлагаете выполнить вашему компьютеру. Понять начало и окончание команды несложно. Вначале указывается имя rsync, затем опции, после них исходный каталог, который содержит файлы для копирования (/home/sam/documents/ на машине coleridge), после этого следует целевой каталог, в который должны быть скопированы файлы (/media/backup/documents на компьютере wordsworth). Перед тем как приступать к рассмотрению опций, надо обратить внимание на особенности указания исходного и целевого каталогов; если не учитывать их, могут возникнуть проблемы.

В данном случае нам надо скопировать содержимое каталога documents, расположенного на машине coleridge,

но не сам каталог, поэтому мы указываем documents/, а не documents. Последняя косая черта в выражении /home/sam/documents/ сообщает команде rsync о том, что необходимо копировать содержимое заданного каталога в каталог documents, расположенный на машине wordsworth. Если вместо documents/ мы зададим documents, будет скопировано и содержимое, и сам каталог, в результате чего на компьютере wordsworth появится каталог /media/backup/documents/documents.

На заметку

Косая черта важна только при указании исходного каталога. Задавая целевой каталог, включать ее не обязательно.

Существует опция, которая не была указана в приведенных выше примерах, но которая может быть очень полезна при использовании команды rsync, — это опция -n (или --dry-run). Если вы зададите эту опцию, команда rsync выполнится, но не будет реально удалять или копировать файлы. Такая "пробная попытка" особенно полезна, когда есть опасность удалить важные файлы. Прежде чем вызывать команду rsync, особенно если для нее указана опция --delete, испробуйте ее "на холостом ходу".

Теперь рассмотрим остальные опции. Опция -v (или --verbose) вместе с опцией --progress указывает команде rsync на то, что в процессе работы надо подробно информировать пользователя о выполненных действиях. Вы видели результаты выполнения команды ранее в этом разделе. В данном случае команда rsync сообщала об удаленных и скопированных файлах. Если вы включаете вызов команды rsync в состав сценария, опция -v, вероятно, будет излишней, но при интерактивном выполнении данной команды полезно иметь сведения о выполняемых действиях. Метаданные в конце информации, отображаемой rsync,

т.е. сведения о числе переданных файлов и их общем размере, а также другие данные выводятся потому, что при вызове команды была указана опция `--stats`. Эта опция, как и `-v`, не нужна при выполнении команды `rsync` в составе сценария, но полезна при вызове команды вручную.

Опция `-r` (или `--recursive`) знакома вам по другим командам. Здесь она имеет то же назначение, что и обычно, — расширяет область действия команды на подкаталоги. В приведенном выше примере эта опция задана, поскольку нам надо скопировать все содержимое каталога `documents`.

Опция `-t` (или `--times`) указывает команде `rsync` на то, что вместе с файлами надо копировать информацию о времени их модификации. Если вы не включите эту опцию, команда `rsync` не сможет определить состояние файлов, скопированных ранее, поэтому при очередном вызове снова скопирует все файлы. Такое поведение вряд ли устроит вас, так как оно сводит на нет все преимущества команды `rsync`. Поэтому всегда указывайте при вызове данной команды опцию `-t`.

Права доступа обсуждались в главе 7. Теперь пришло время снова вспомнить о них. Опция `-p` (или `--perms`) указывает `rsync` на то, что для каждого файла, записанного в целевой каталог, надо установить те же права доступа, что и для исходного файла. Данная опция позволяет выполнять резервное копирование максимально точно, поэтому не следует пренебрегать ею.

Если в исходном каталоге присутствовала ссылка, опция `-l` (или `--links`) создает такую же ссылку в целевом каталоге. Вместо копирования файла копируется ссылка на него. Таким образом, содержимое каталога воссоздается в том виде, в каком оно было на компьютере-источнике.

Даже при наличии быстродействующих соединений желательно использовать опцию `-z` (или `--compress`). Если

она указана, то при передаче файлов rsync выполняет gzip-сжатие. Если линии связи имеют малую пропускную способность, эта опция обязательна. Для высокоскоростных линий также желательно задавать ее, так как она позволит сэкономить вам время.

Обеспечить безопасность позволяет опция -e (или --rsh=ssh). Она задает передачу данных средствами ssh. Как видите, команда rsync обеспечивает не только быстрое, но и безопасное копирование данных.

На заметку

Если используется система SSH, то почему не задается пароль? Ранее в этой главе мы настроили средства SSH для работы без введения пароля.

И напоследок рассмотрим самую опасную опцию --delete. Если вы создаете зеркальное отражение файлов, то, вероятно, хотите, чтобы оно было как можно более точным. Это означает, что файлы, удаленные в исходном каталоге, должны быть удалены и в целевом. Но при этом можно случайно удалить нужные данные. Если вы собираетесь задать опцию --delete (а рано или поздно вам придется сделать это), сначала выполните команду с опцией -n (или --dry-run), которая рассматривалась выше.

В команде rsync предусмотрено много других опций; существует множество вариантов ее применения (в справочном руководстве описывается восемь таких вариантов). Материал этого раздела позволит вам начать работу с команды rsync. Вызовите на своем компьютере команду man rsync или выполните средствами Google поиск по ключевым словам rsync tutorial, и вы найдете огромное количество информации о команде rsync. Если вы случайно удалите файл, а потом вспомните, что существует его резервная копия, созданная с помощью rsync, вы поймете,

что не зря потратили время на изучение этой замечательной команды.

Совет

Если вы хотите, чтобы ваши данные действительно были в безопасности, организуйте регулярный запуск rsync средствами cron. Например, создайте файл backup.sh (его можно поместить в каталог ~/bin) и введите в нем следующую команду:

```
$ rsync --verbose --progress -stats --recursive --times
--perms -links --compress --rsh=ssh -delete /home/sam/
documents/ will@wordsworth:/media/backup/documents
```

С помощью chmod установите признак исполняемого файла.

```
$ chmod 744 /home/scott/bin/backup.sh
```

Затем включите в файл cronfile (он также может храниться в каталоге ~/bin) следующие строки:

```
# backup documents every morning at 3:05 am
05 03 * * * /home/scott/bin/backup.sh
```

Первая строка представляет собой комментарии, в которых объясняется назначение задачи, а вторая строка сообщает cron о том, что каждую ночь в 3:05 надо автоматически выполнить инструкции, заданные в файле /home/scott/bin/backup.sh.

После создайте задачу для cron.

```
$ crontab /home/scott/bin/cronfile
```

Теперь вам не надо заботиться о создании резервных копий. Требуемые действия будут выполняться автоматически, вам надо лишь оставить компьютер включенным на ночь.

(Дополнительную информацию о cron можно получить, выполнив команду man cron либо обратившись к документу *Newbie: Intro to cron*, который доступен по адресу http://www.unixgeeks.org/security/newbie/unix/cron-1.html.)

Копирование файлов из Web

`wget`

Глобальная сеть — неисчерпаемый источник данных. В частности, в ней есть огромное количество изображений, видео- и аудиофайлов, доступных для копирования. Однако если вручную копировать, например, каждый mp3-файл из каталога, содержащего 200 таких файлов, эта затея быстро надоест. Команда `wget` позволяет организовать копирование файлов с Web-узлов, но не предоставляет графического пользовательского интерфейса. Вы вводите ее в командной строке, и спустя некоторое время (возможно, даже через несколько часов) она предоставит вам результат.

Однако правильно вызвать команду `wget` не всегда просто. Она предоставляет огромные возможности, описание которых займет отдельную книгу, и в рамках одной главы невозможно рассказать, что она позволяет сделать. Поэтому мы сосредоточимся на решении с помощью команды `wget` двух задач: копирования множества отдельных файлов и копирования целых Web-узлов.

В сети есть узел The Old Time Radio Archives, на котором собраны записи радиопередач, представленных в формате mp3. Годовое собрание составляет 365 mp3-файлов, по одному файлу на каждый день в году. Предположим, вы хотите скопировать эти файлы, но перспективу щелкать правой кнопкой мыши на каждой ссылке, выбирать в контекстном меню пункт меню Save Link As ..., а затем щелкать на кнопке OK, вряд ли можно назвать привлекательной.

Ознакомившись с каталогом, вы заметите, что он организован следующим образом:

```
http://www.oldtimeradioarchives.com/mp3/
  season_10/
  season_11/
```

```
...
season_20/
...
season_3/
season_4/
...
season_9/
```

На заметку

Каталоги отсортированы не по номерам, как это сделал бы практически каждый человек, а в алфавитном порядке. Именно такой принцип сортировки по умолчанию реализует большинство компьютерных программ. В этом случае получается, что слово "ten" предшествует слову "three".

В каждом каталоге размещены mp3-файлы. Если вы щелкнете на ссылке на каталог, отобразится Web-страница, в которой перечислены файлы из этого каталога.

```
[BACK] Parent Directory   19-May-2002 01:03 -
[SND]  1944-12-24_532.mp3 06-Jul-2002 13:54 6.0M
[SND]  1944-12-31_533.mp3 06-Jul-2002 14:28 6.5M
[SND]  1945-01-07_534.mp3 06-Jul-2002 20:05 6.8M
```

В данном случае последовательность символов [SND] заменяет изображение нот, которое выводится перед именем каждого файла в списке.

Таким образом, задача сводится к следующему: как скопировать все mp3-файлы, имеющие различные имена и находящиеся в разных каталогах. Решить ее поможет команда wget.

Начните с создания на вашем компьютере каталога, в который вы поместите скопированные mp3-файлы.

```
$ mkdir radio_mp3s
```

Затем сделайте этот каталог текущим и вызовите команду wget.

```
$ cd radio_mp3s
$ wget -r -l2 -np -w 5 -A.mp3 -R.html,.gif
  http://www.oldtimeradioarchives.com/mp3/
```

Рассмотрим подробнее команду и ее опции.

В начале командной строки указано имя команды, а в конце — URL ресурса, с которым она должна работать: `http://www.oldtimeradioarchives.com/mp3`. Между именем команды и URL содержатся опции, которым необходимо уделить особое внимание.

Опция `-r` (или `--recursive`) задает следование по ссылкам и просмотр каталогов в поисках файлов. В данном случае рекурсивное выполнение `wget` означает, что команда просмотрит каталоги, соответствующие каждому сезону, и скопирует найденные в них mp3-файлы.

Опция `-l2` (или `--level=[#]`) очень важна. Она указывает команде `wget` глубину рекурсивного поиска файлов. Буква `l` означает `level`, или уровень, а следующее за ней число — глубину поиска. Если вы введете `-l1`, команда `wget` будет извлекать файлы только из каталога `/mp3`. В результате вы ничего не получите, так как в этом каталоге находятся лишь подкаталоги `season_10`, `season_11` и т.д., а уже в них размещаются mp3-файлы. Задав опцию `-l2`, вы сообщите команде `wget` о том, что надо в первую очередь проверить `/mp3` (этот каталог соответствует первому уровню), а затем по очереди обратиться к каждому каталогу `season_#` и извлечь файлы из него. Задавая уровень, надо быть очень внимательным, так как вы можете заполнить весь жесткий диск ненужными данными.

Один из способов избежать копирования лишних файлов — использование опции `-np` (или `--no-parent`). Она не позволяет осуществлять рекурсивный поиск в роди-

тельском каталоге. Если вы вернетесь к предыдущему листингу, то заметите, что первая ссылка указывает именно на родительский каталог. Если вы находитесь в каталоге season_10, то родительским для него является /mp3. То же самое справедливо для season_11, season_12 и т.д. Однако нам не нужно перемещение вверх по иерархии объектов, команда wget должна перемещаться только вниз. И конечно же, не следует тратить время на возврат в уже обработанный каталог, в данном случае /mp3.

Следующая опция не обязательна, однако, если вы не хотите доставить лишние хлопоты администратору узла, к которому вы обращаетесь, установите ее. Речь идет об опции -w (или --wait=[#]), которая задает небольшую задержку перед началом копирования каждого файла. Это предотвращает излишнюю нагрузку на сервер, возникающую вследствие постоянного извлечения файлов. По умолчанию считается, что значение указано в секундах, но при желании вы можете использовать задержку в минутах, если введете после числа букву m, в часах (буква h) и даже в днях (буква d).

Теперь начинается самое интересное. Опция -A (или --accept) сообщает команде wget о том, что вы хотите скопировать только файлы определенного типа и никакие другие. В данном случае A означает accept; после опции указываются суффиксы имен файлов, разделенные запятыми. Нам нужны файлы только одного типа, mp3, поэтому мы задаем -A.mp3.

Противоположные действия выполняет опция -R (или --reject). С ее помощью мы указываем файлы, которые нам не нужны, а именно HTML и GIF. Таким образом мы, в частности, отказываемся от пиктограммы с изображением нот, которая в приведенном выше листинге представлялась с помощью символов [SND]. Суффиксы в списке разделяются запятыми, и опция имеет следующий вид: -R.html,.gif.

При выполнении команды wget с указанным выше набором опций на ваш компьютер будет скопировано 365 mp3-файлов. Не исключено, что передача прервется. Это может произойти по многим причинам: маршрутизатор выйдет из строя, или кто-то наступит на сетевой кабель и выдернет его из гнезда, или при монтажных работах будет поврежден волоконно-оптический кабель и т.д. В этом случае вам надо повторно вызвать команду, задав дополнительно опцию -c (или --continue), которая сообщает wget о том, что надо возобновить прерванное копирование. В этом случае уже полученные файлы не будут копироваться вновь.

Рассмотрим еще один пример использования команды wget для копирования файлов. Предположим, что мы узнали о появлении двух новых альбомов. Файлы mp3 с композициями из этих альбомов перечислены на странице http://www.djbc.net/beastles/. Приведенная ниже команда извлечет ссылки с Web-страницы, запишет их в файл, а затем инициирует копирование соответствующих ресурсов средствами wget.

```
$ dog --links http://www.djbc.net/beastles/ | grep
mp3 > beastles ; wget -i beastles
--12:58:12-- http://www.djbc.net/beastles/
webcontent/djbc-holdittogethernow.mp3
           => 'djbc-holdittogethernow.mp3'
Resolving www.djbc.net... 216.227.209.173
Connecting to www.djbc.net|216.227.209.173|:80...
connected.
HTTP request sent, awaiting response... 200 OK
Length: 4,533,083 (4.3M) [audio/mpeg]
100%[==========>] 4,533,083 203.20K/s ETA 00:00
12:58:39 (166.88 KB/s) - 'djbc-holdittogethernow.
mp3' saved [4533083/4533083]
```

В главе 5 мы говорили о команде `cat`, которая выводит содержимое файлов в стандартный выходной поток. Там же шла речь о дальнейшем развитии данной команды — программе `dog`. Если вы вызовете `dog` с опцией `--links` и зададите ей в качестве параметра URL, ссылки будут извлечены из указанного документа и выведены в `STDOUT`. В данном примере полученные ссылки передаются команде `grep` (см. главу 9), которая фильтрует данные, оставляя строки, содержащие символы `mp3`, а затем выходная информация перенаправляется в текстовый файл `beastles` (о перенаправлении вывода см. в главе 4).

После точки с запятой вводится новая команда — `wget` (о разделении команд посредством точки с запятой см. в главе 4). Опция `-i` (или `--input-file`) сообщает команде `wget` о том, что URL ресурсов для копирования указываются не в стандартном входном потоке, а в файле. Если у вас есть много ссылок, поместите их в файл, а затем задайте при вызове `wget` опцию `-i`. В данном примере эта опция указывает на файл `beastles`, созданный в результате совместной работы команд `dog` и `grep`, и `wget` начинает копировать требуемые файлы один за другим.

И все эти действия были заданы в одной строке. Такие богатые возможности может предоставить только Linux!

Копирование Web-узлов

```
wget
```

Если вы хотите скопировать на свой компьютер Web-узел или создать резервную копию своего узла, вам поможет в этом команда `wget`. В предыдущем разделе мы использовали команду `wget` для получения отдельных файлов, но она позволяет также работать и с целыми узлами.

На заметку

Используйте команду wget разумно. Не пытайтесь скопировать узлы, содержащие слишком большой объем данных, и имейте в виду, что узел, который вы собираетесь скопировать, кто-то создал и имеет права на него.

Предположим, что вас интересует узел `http://www.neato.com`, а сами вы в текущий момент работаете с документом `http://www.neato.com/articles/index.htm`. Вам нужно содержимое раздела `/articles` и ничего более. Приведенная ниже команда предоставит вам требуемые данные.

```
$ wget -E -r -k -p -w 5 -np
  http://www.neato.com/articles/index.htm
```

При желании вы можете объединить опции так, как показано ниже.

```
$ wget -Erkp -w 5 -np
  http://www.neato.com/articles/index.htm
```

Подобно примеру из предыдущего раздела, данная строка начинается с имени wget и заканчивается интересующим нас URL. Вы уже знакомы с опциями `-w` (или `--wait=[#]`), `-np` (или `--no-parent`) и `-r` (или `--recursive`); здесь они выполняют те же функции, что и прежде. Рассмотрим подробнее опции, которые встретились вам впервые.

На узле, предназначенном для копирования, не все файлы имеют суффиксы `.htm` или `.html`; встречаются также файлы `.asp`, `.php`, `.cfm` и т.д. Если вы попытаетесь просмотреть содержимое такого узла, скопированного на свой компьютер, возникнут проблемы. Если на вашей машине установлен Web-сервер, все в порядке, но вероятнее всего, что его там нет. Однако и без сервера файлы `.htm` и `.html` можно открыть с помощью браузера. Если вы зададите

опцию -E (или --html-extension), команда wget преобразует каждую страницу в html-файл, что позволит вам просматривать их с помощью браузера, не используя дополнительные программы.

При копировании узла возникают и другие проблемы, с которыми можно успешно справиться, правильно выбрав набор опций команды wget. Не исключено, что, открыв Web-документ на своем компьютере, вы обнаружите, что ссылки не работают и вы не можете переходить с одной страницы на другую. Задав опцию -k (или --convert-links), вы укажете команде wget на то, что надо преобразовать ссылки в составе Web-страниц. Эта опция затрагивает ссылки, указывающие не только на другие страницы, но и на изображения, каскадные таблицы стилей и файлы. Используйте ее, и вы будете довольны результатами.

Наличие каскадных таблиц стилей (Cascading Style Sheets — CSS) и изображений обусловливает необходимость установить опцию -p (или --page-requisites). Для того чтобы Web-страница отображалась корректно, Web-разработчик должен включать в нее ссылки на файлы, содержащие таблицы стилей. изображения и JavaScript-сценарии, которые будут использоваться с HTML-файлом. Опция -p указывает команде wget на то, что следует скопировать все файлы, необходимые для отображения Web-страницы. Если данная опция задана, то, открыв на своем компьютере скопированную Web-страницу, вы увидите, что она выглядит точно так же, как будто бы вы открыли ее, обращаясь к серверу. Без этой опции вы обнаружите, что некоторые файлы отсутствуют.

Страница справочного руководства, посвященная команде wget, содержит огромный объем информации. Освоив ее, вы сможете использовать данную команду для решения еще более сложных задач. Если, прочитав данную главу, вы поняли, что команда wget будет полезна вам, принимайтесь за изучение руководства.

Указание последовательностей имен копируемых файлов

```
curl
```

На первый взгляд команды `wget` и `curl` выглядят одинаково: обе позволяют копировать файлы при запуске из командной строки. Однако между ними есть одно существенное различие и много мелких. Главное отличие состоит в том, что команда `curl` позволяет указывать последовательности и множества файлов, предназначенных для копирования, а команда `which` не позволяет сделать этого; с другой стороны, `wget` поддерживает рекурсивную обработку, чего не обеспечивает `curl`.

На заметку

Эти программы имеют множество более мелких отличий. Полный список возможностей команды `curl` можно найти в документе *Features – what can curl do* (`http://curl.haxx.se/docs/features.html`), а некоторые функции `wget` описаны в документе *Overview* (`http://www.gnu.org/software/wget/manual/html_node/Overview.html#Overview`). На узле, посвященном программе `curl`, содержится диаграмма сравнения `curl` с другими продуктами аналогичного назначения. Она включена в документ *Compare cURL Features with Other FTP+HTTP Tools* (`http://curl.haxx.se/docs/comparison-table.html`). Судя только по диаграмме, предпочтение надо отдавать `curl` (это неудивительно, учитывая направленность материалов на узле).

Рассмотрим пример, демонстрирующий способность команды `curl` поддерживать последовательности имен файлов, предназначенных для копирования. Public Radio предоставляет архивы записей своей передачи This American Life. Эти архивы доступны для копирования с Web-узла в формате Real Audio (почему в данном случае выбран

данный формат, а не другой, открытый, остается для меня загадкой). Если вы хотите скопировать десять файлов Real Audio, вам надо ввести следующую команду:

```
$ curl -O http://www.wbez.org/ta/[1-10].rm
[1/10]: http://www.wbez.org/ta/1.rm --> 1.rm
--_curl_--http://www.wbez.org/ta/1.rm
```

Обратите внимание: для того, чтобы указать, что вы хотите скопировать файлы `1.rm`, `2.rm`, `3.rm` и т.д., вы используете запись `[1-10].rm`. Если бы эти файлы имели имена `one.rm`, `two.rm` и `three.rm`, вам пришлось бы использовать вместо последовательности множество.

```
$ curl -O http://www.wbez.org/ta/{one,two,three}.rm
```

Опция `-O` (или `--remote-name`) необходима. Если вы не будете использовать ее, команда `curl` выведет результаты копирования в стандартный выходной поток, в результате чего окно вашего терминала будет заполнено непонятными символами. Опция `-O` указывает команде `curl` на то, что данные надо записывать в файл и что имя создаваемого файла на локальном компьютере должно совпадать с именем соответствующего файла на удаленной машине.

В данном разделе были приведены лишь самые общие сведения о команде `curl`. Страница справочного руководства, посвященная этой команде, имеет меньший объем, чем страница о команде `wget`, но также предоставляет полезную информацию, которая позволит вам в полной мере использовать возможности `curl`. Советую прочитать ее.

Выводы

Команды, описанные в этой главе, выбраны по одному принципу: лично я предпочитаю их остальным, так как они помогают мне решить трудные задачи, требующие выпол-

нения рутинной работы либо представляющие опасность для информации на компьютере. Эти команды расширяют сферу применения данных, программ, да и вашего воображения на всю сеть. Предоставляя свободный доступ к таким мощным инструментам, как `ssh`, `rsync`, `wget` и `curl`, Linux открывает дорогу для использования новых технологий. В операционных системах, которые почему-то предпочитают большинство пользователей, это не всегда возможно. Если вы хотите устанавливать защищенные соединения с другими компьютерами, освойте средства `ssh`. Чтобы автоматически выполнять резервное копирование, обеспечивая при этом безопасность данных, изучите команду `rsync`. Программы `wget` и `curl` позволят вам наиболее эффективно копировать требуемые ресурсы. Эти инструменты не подведут вас!

16

Взаимодействие с системой Windows

Samba — один из самых важных проектов с открытым кодом, поскольку он предназначен для обеспечения поддержки в системе Linux (а также в Unix и Mac OS X) сетевого протокола SMB (Session Message Block), который применяется всеми машинами, работающими под управлением Microsoft Windows. Samba позволяет связать машины, использующие систему Unix, с Windows-машинами, монтировать диски, организовывать их совместное использование, выводить данные на разделяемые принтеры, подключенные к Windows-компьютерам. Можно также установить на машине под управлением Unix принтер и создать файлы, к которым будут обращаться компьютеры, работающие под Windows. По сути, система Samba может использоваться даже при отсутствии Windows-машин. С помощью этой системы можно организовать разделение файлов и принтеров между компьютерами, на которых установлена система Linux.

В настоящее время вопросам установки Samba на сервере посвящено много книг. Здесь мы не будем рассматривать команды администрирования, такие как `smbd`, `smbcacls` или `smbpasswd`, или обсуждать вопросы изменения содержимого файла `smb.conf`. (Несмотря на то что

перечисленные выше команды могут применять обычные пользователи, например, для того, чтобы изменять свои пароли, на практике их обычно используют лишь системные администраторы.) При изложении материала данной главы предполагается, что средства SMB уже установлены на вашем компьютере, поэтому основное внимание будет уделено работе клиента. Мы рассмотрим, как находить разделяемые ресурсы, как устанавливать соединение с ними и как монтировать их.

Совет

Для тех, кто интересуется вопросами администрирования Samba, можно порекомендовать несколько книг.

- Стив Литт (Steve Litt) *Sams Samba Unleashed*, ISBN: 0672318628.
- Джеральд Картер (Gerald Carter) *Sams Teach Yourself Samba in 24 Hours, 2nd Edition*, ISBN: 0672322692.
- *The Official Samba-3 HOWTO and Reference Guide* `http://samba.org/samba/docs/man/Samba-HOWTO-Collection/`.

Обнаружение Master Browser рабочей группы

```
nmblookup -M [Master Browser]
nmblookup -S [имя NetBIOS]
nmblookup -A [IP-адрес]
```

Для организации работы сервера Samba используются два демона: `smbd`, обеспечивающий доступ к разделяемым ресурсам, и `nmbd`, отображающий имена NetBIOS, которые идентифицируют машины, использующие SMB, в IP-адреса; благодаря этому становится возможным находить и просматривать разделяемые ресурсы. В данном разделе мы рассмотрим команды, взаимодействующие с `nmbd` и предоставляющие общую информацию о рабочей группе Windows.

На заметку

В последующих примерах предполагается, что мы используем рабочую группу Windows, а не домен. Рабочая группа — это несколько машин, которые идентифицируют себя как членов группы. Домен использует центральный сервер или, в больших сетях, несколько серверов, выполняющих аутентификацию компьютеров и пользователей, подключающихся к сети. Домены имеют большой объем и сложны в управлении; они подробно рассматриваются в книгах, ссылки на которые приведены в начале главы. Если вы хотите изучить домены, обратитесь к этим источникам.

В рабочей группе Windows вам нужна машина, которая следит за остальными членами группы, в частности, контролирует их SMB-имена и IP-адреса. Эта машина называется Master Browser. Но какой компьютер в группе выполняет эти функции? Он определяется в результате "голосования". Критерием для его выбора является используемая на нем операционная система. Преимущество отдается самой мощной системе, в частности ее последней версии. Таким образом, Windows XP будет иметь преимущество перед Windows 2000, а она, в свою очередь, "победит" Windows 98.

Настраивая сервер Samba, можно сконфигурировать его так, что он не будет участвовать в "голосовании", предоставляя это право другим машинам, либо так, что он всегда будет выигрывать "голосование". Если вы знаете о разделяемых ресурсах Samba, то можете устанавливать соединения с ними, но если у вас есть какие-либо сомнения, надо найти Master Browser.

Для того чтобы запросить по сети информацию о Master Browser, вызовите команду `nmblookup` с опцией `-M` (или `--master-browser`), задав в конце строки символ `-`. Он означает "найти Master Browser". Однако при этом возникает проблема: вы не можете задавать символ - в командной строке, так как оболочка интерпретирует его как начало

опции. Выход — указать перед этим символом знаки --; этим вы сообщите оболочке, что следующий символ - не является частью опции.

```
$ nmblookup -M -- -
querying __MSBROWSE__ on 192.168.1.255
192.168.1.151 __MSBROWSE__<01>
192.168.1.104 __MSBROWSE__<01>
```

Результаты не очень хорошие. В данном случае оказывается, что в сети присутствуют два Master Browser. Это может стать проблемой, поскольку разные Master Browser в разное время имеют сведения о различных машинах, что существенно затрудняет работу. В один момент информация о пользователе может быть на машине, а в другой момент отсутствовать. Почему так происходит? Дел в том, что один Master Browser знает о машине, а другой — нет.

Для того чтобы получить дополнительную информацию о Master Browser, вызовите команду `nmblookup` с опцией `-S` (или `--status`), которая возвращает SMB-имена, используемые узлами сети.

```
$ nmblookup -S 192.168.1.151
querying 192.168.1.151 on 192.168.1.255
name_query failed to find name 192.168.1.151
```

Как видите, результат не получен, поскольку опция `-S` требует имя NetBIOS, а не IP-адрес. К сожалению, мы не знаем имя машины, а только IP-адрес. На самом деле проблемы в этом нет. Вам надо лишь добавить опцию `-A` (или `--lookup-by-ip`), которая указывает `nmblookup` на то, что вместо имени NetBIOS мы передаем ей IP-адрес.

```
$ nmblookup -SA 192.168.1.151
Looking up status of 192.168.1.151
        JANSMAC         <00> -          B <ACTIVE>
```

```
        JANSMAC          <03> -            B <ACTIVE>
        JANSMAC          <20> -            B <ACTIVE>
        ..__MSBROWSE__.  <01> - <GROUP>    B <ACTIVE>
        MILTON           <00> - <GROUP>    B <ACTIVE>
        MILTON           <1d> -            B <ACTIVE>
        MILTON           <1e> - <GROUP>    B <ACTIVE>

        MAC Address = 00-00-00-00-00-00
```

Теперь мы знаем, что компьютер по адресу 192.168.1.151 идентифицируется как JANSMAC (он должен работать под управлением Mac OS) и является Master Browser для рабочей группы MILTON. А как обстоит дело для машины, имеющей другой IP-адрес?

```
$ nmblookup -SA 192.168.1.104
Looking up status of 192.168.1.104
        ELIOT            <00> -            B <ACTIVE>
        ELIOT            <03> -            B <ACTIVE>
        ELIOT            <20> -            B <ACTIVE>
        ..__MSBROWSE__.  <01> - <GROUP>    B <ACTIVE>
        TURING           <00> - <GROUP>    B <ACTIVE>
        TURING           <1d> -            B <ACTIVE>
        TURING           <1e> - <GROUP>    B <ACTIVE>

        MAC Address = 00-00-00-00-00-00
```

Компьютер по адресу 192.168.1.104 имеет NetBIOS-имя ELIOT и является Master Browser для рабочей группы TURING. Таким образом, нам беспокоиться не о чем, так как каждая из двух машин является Master Browser для своей рабочей группы. Компьютеры, которые считают себя членами группы MILTON, будут обращаться за информацией к JANSMAC, а компьютеры, принадлежащие группе TURING, будут использовать для этой цели компьютер ELIOT.

На заметку

Дополнительную информацию, позволяющую лучше понять значение данных, выведенных в предыдущих примерах, можно получить, обратившись к документу, расположенному по адресу `http://support.microsoft.com/kb/q163409/`.

Запрос имен NetBIOS и IP-адресов

```
nmblookup -T
```

Команда `nmblookup` позволяет быстро находить компьютеры, совместно использующие файлы и принтеры, посредством Samba. Задайте при вызове команды опцию `-T`, введите после нее последовательность символов `"*"` (в данном случае звездочка должна быть помещена в кавычки, чтобы оболочка не интерпретировала ее как символ групповой операции, представляющий файлы в текущем каталоге), и вы получите результаты, подобные приведенным ниже.

```
$ nmblookup -T "*"
querying * on 192.168.1.255
192.168.1.151 *<00>
192.168.1.104 *<00>
192.168.1.10 *<00>
```

В локальной сети есть три машины с разделяемыми ресурсами Samba. Как определить, какая из них является Master Browser, описано в предыдущем разделе, а в следующем разделе пойдет речь о получении списка разделяемых ресурсов, предлагаемых каждым компьютером.

Получение списка разделяемых ресурсов

```
smbclient
```

В результате выполнения описанной выше команды мы узнали о том, что на некоторых машинах есть разделяемые ресурсы Samba, но нам нужна более подробная информация о них. Команда `smbclient` позволяет устанавливать соединения с этими ресурсами и работать с ними. Она также дает возможность получить список разделяемых ресурсов, доступных на том или ином компьютере. Для решения этой задачи используется опция `-L` (или `--list`), за которой указывается имя NetBIOS или IP-адрес. В данном примере в ответ на приглашение для ввода пароля достаточно нажать клавишу <Enter>.

```
$ smbclient -L ELIOT
Password:
Anonymous login successful
Domain=[TURING] OS=[Unix] Server=[Samba 3.0.14a-
Ubuntu]
Sharename Type Comment

print$    Disk Printer Drivers
documents Disk Shared presentations and other
files
IPC$      IPC  IPC Service (eliot server (Samba,
Ubuntu))
ADMIN$    IPC  IPC Service (eliot server (Samba,
Ubuntu))
Anonymous login successful
Domain=[TURING] OS=[Unix]
Server=[Samba 3.0.14a-Ubuntu]
```

Мы видим все разделяемые ресурсы, доступные пользователю, зарегистрировавшемуся анонимно, или поме-

ченные как разрешенные для просмотра в файле smb.conf на сервере. Для того чтобы увидеть ресурсы, которые будут доступны вам после регистрации, добавьте опцию -U (или --user) и укажите после нее ваше имя пользователя Samba.

На заметку

Имя пользователя Samba не обязательно должно совпадать с регистрационным именем в системе Linux (или Windows или Mac OS X) на сервере Samba.

```
$ smbclient -L ELIOT -U scott
Password:
Domain=[ELIOT] OS=[Unix]
Server=[Samba 3.0.14a-Ubuntu]
Sharename Type Comment

print$    Disk Printer Drivers
documents Disk Shared presentations and other
files
IPC$      IPC  IPC Service (eliot server (Samba,
Ubuntu))
ADMIN$    IPC  IPC Service (eliot server (Samba,
Ubuntu))
scott Disk Home Directories
Domain=[ELIOT] OS=[Unix]
Server=[Samba 3.0.14a-Ubuntu]
```

После регистрации вы увидите новый разделяемый ресурс scott — рабочий каталог пользователя. Это сообщение также является признаком того, что вы можете зарегистрироваться. О регистрации и использовании ресурсов речь пойдет в следующем разделе.

Совет

Если вы хотите проверить разделяемые ресурсы, созданные вами на сервере Samba, это можно сделать, указав в оболочке на компьютере, содержащем ресурсы, следующую команду:

```
$ smbclient -L localhost
```

В ответ на приглашение ввести пароль нажмите клавишу <Enter>. Таким образом, вы сможете быстро выяснить, доступны ли новые ресурсы, созданные вами. Если вы не разрешили просмотр ресурсов, вам надо зарегистрироваться как пользователь, который имеет право на работу с ними.

Обращение к ресурсам Samba с помощью FTP-подобного клиента

`smbclient`

Зная, что разделяемые ресурсы доступны на сервере Samba, можно зарегистрироваться и начать работу с ними. Для этого используется та же команда `smbclient`, но вызывается она в следующем формате:

```
smbclient //сервер/разделяемый_ресурс -U
  имя_пользователя
```

Таким образом нельзя зарегистрироваться на сервере, можно лишь зарегистрироваться для пользования ресурсом. При обращении к ресурсам, защищенным паролем (а использовать пароль рекомендуется), надо указать имя пользователя. Например, для доступа к ресурсу `documents` на сервере `ELIOT` следует использовать следующую команду:

```
$ smbclient //eliot/documents -U scott
Password:
Domain=[ELIOT] OS=[Unix]
Server=[Samba 3.0.14a-Ubuntu]
smb: \>
```

Об успешном выполнении свидетельствует подсказка для ввода команд Samba, которая имеет вид `smb: \>`. Заметьте, что вам предлагается ввести пароль. Предположим, что для пользователя `scott` пароль имеет вид `123456` (в реальной работе подобные пароли использовать не рекомендуется, но это лишь пример). Тогда если вы введете команду так, как показано ниже, пароль запрашиваться не будет.

```
$ smbclient //eliot/documents -U scott%123456
```

Поступать так нежелательно, поскольку каждый, кто сможет просмотреть ваш файл `.bash_history` (о предыстории выполнения команд см. в главе 11) или список выполняемых процессов (см. главу 12), узнает ваш пароль. Никогда не присоединяйте пароль к пользовательскому имени! С точки зрения безопасности гораздо лучше будет, если вы введете его в ответ на приглашение.

Совет

Если вы пишете сценарий и интерактивный режим регистрации не подходит вам, все равно не стоит вводить пароль после пользовательского имени. Вместо этого надо использовать опцию `-A` (или `--authentication-file=`*имя_файла*), которая ссылается на файл с конфиденциальными данными. Для пользователя `scott` этот файл должен содержать следующую информацию:

```
username = scott
password = 123456
```

С помощью команды `chmod` (см. главу 7) установите такие права доступа к этому файлу, которые не позволят посторонним прочитать его содержимое.

Установив соединение с разделяемым ресурсом Samba, вы можете использовать команды, хорошо знакомые тем, кто использует клиентскую программу `ftp`, запускаемую из командной строки. Список этих команд приведен в табл. 16.1.

Таблица 16.1. Наиболее часто используемые команды `smbclient`

Команда	Описание
`cd`	Изменить текущий каталог
`exit`	Закрыть соединение с сервером Samba
`get`	Скопировать указанный файл на локальную машину
`help`	Отобразить справочную информацию
`lcd`	Изменить текущий каталог на локальной машине
`ls`	Получить список файлов, содержащихся в текущем каталоге на сервере Samba
`mget`	Скопировать файлы, имена которых соответствуют шаблону, на локальную машину
`mkdir`	Создать новый каталог на сервере Samba
`mput`	Скопировать файлы, имена которых соответствуют шаблону, на сервер Samba
`put`	Скопировать указанные файлы на сервер Samba
`rm`	Удалить указанные файлы с сервера Samba

Существуют и другие команды, получить информацию о которых вы можете по команде `help`. В табл. 16.1 представлены лишь те из них, которые используются наиболее часто. Закончив работу, введите `exit`. Этим вы завершите сеанс взаимодействия с сервером и вернетесь в оболочку на вашей машине.

Монтирование файловой системы Samba

```
smbmount
smbumount
```

Команда `smbclient` обеспечивает доступ к разделяемым ресурсам Samba, но она не подходит, если вам надо

пользоваться программами с графическим интерфейсом либо выполнять много работы с разделяемым ресурсом. В этих случаях надо использовать команду smbmount, которая монтирует ресурс Samba в локальной файловой системе. В результате вы сможете взаимодействовать с этим ресурсом так же, как и со своим жестким диском. После монтирования вы можете использовать любую программу для работы с файлами и взаимодействовать с ее помощью с разделяемым ресурсом. Можно даже обращаться к ресурсу из прикладных программ, например из текстового процессора, и непосредственно открывать файлы.

Для того чтобы смонтировать разделяемый ресурс Samba в файловой системе своего компьютера, вам надо сначала выбрать точку монтирования. Например, если вы собираетесь монтировать ресурс documents, расположенный на сервере ELIOT, можете создать следующий каталог:

```
$ mkdir /home/scott/eliot_documents
```

Теперь можно вызывать команду smbmount и задавать при вызове ее опции.

```
$ smbmount //eliot/documents /home/scott/
eliot_documents -o credentials=/home/scott/bin/
credentials_eliot,fmask=644,dmask=755,uid=1001,gi
d=1001,workgroup=TURING
smbmnt must be installed suid root for direct user
mounts (1000,1000)
smbmnt failed: 1
```

Прежде чем говорить об опциях, выясним, почему при выполнении команды smbmount возникла ошибка. Во второй строке листинга сказано, что программа smbmnt, вызываемая smbmount, может быть запущена только от имени пользователя root. Однако в процессе работы обычные пользователи также должны иметь возможность монтиро-

вать ресурсы Samba. Решить проблему можно, установив признак suid (см. главу 7).

На заметку

Вы предоставляете обычным пользователям возможность вызывать smbmnt от имени пользователя root, но это не опасно. Гораздо хуже было бы, если бы вам пришлось дать им пароль root или если бы они обращались к вам каждый раз, когда им надо смонтировать разделяемый ресурс Samba.

Признак suid для smbmnt устанавливается следующим образом:

```
# ls /usr/bin/smbmnt
-rwxr-xr-x 1 root root /usr/bin/smbmnt
# chmod u+s /usr/bin/smbmnt
# ls -l /usr/bin/smbmnt
-rwsr-xr-x 1 root root /usr/bin/smbmnt
```

На заметку

Для экономии места некоторая информация, обычно отображаемая по команде ls -l, здесь не приводится.

Теперь попробуем снова вызвать команду smbmount:

```
$ smbmount //eliot/documents
  /home/scott/eliot_documents -o
  redentials=/home/scott/credentials_eliot,
  fmask=644,dmask=755,uid=1001,gid=1001,
  workgroup=TURING
$ ls -F /home/scott/eliot_documents
presentations/ to_print/
```

Рассмотрим подробнее структуру команды. После имени smbmount следует выражение //eliot/documents. Оно

определяет сервер Samba и разделяемый ресурс, с которым устанавливается соединение. Затем указывается путь к точке монтирования, в данном случае /home/scott/eliot_documents. Опция -o указывает на начало набора опций.

Вместо credentials=/home/scott/credentials_eliot можно воспользоваться одним из следующих выражений:

- username=scott,password=123456
- username=scott%123456
- username=scott

Первое и второе выражения недопустимы с точки зрения безопасности: пароль не должен записываться в файл .bash_history или быть доступным по команде ps. Последнее выражение приведет к тому, что на экране отобразится приглашение для ввода пароля. Такой подход обеспечивает достаточный уровень защиты, но мы стараемся автоматизировать процесс монтирования разделяемых ресурсов Samba, поэтому ввод пароля вручную также не подходит нам.

Таким образом, остается единственное решение: использовать выражение credentials=/home/scott/credentials_eliot. Оно указывает команде smbmount на то, что пользовательское имя и пароль содержатся в файле в следующем формате:

```
username = scott
password = 123456
```

О таком же файле шла речь предыдущем разделе. Так же как и тогда, нам необходимо использовать команду chmod, чтобы ограничить круг пользователей, которые могли бы ознакомиться с содержимым файла. Если вам не надо автоматизировать процесс регистрации, можно использовать выражение username=scott и вводить пароль вруч-

ную. Такое решение, несомненно, наилучшее с точки зрения защиты.

Опции `fmask` и `dmask` управляют назначением прав по умолчанию для новых файлов и каталогов, которые создаются в смонтированном ресурсе Samba. Значение 644 для файлов соответствует набору прав `rw-r--r--`, а значение 755 представляет для каталогов набор прав `rwxr-xr-x`.

На заметку

Если вам непонятны приведенные выше термины, прочитайте еще раз главу 7.

Как вы помните, все имена пользователей и групп соответствуют номерам, которые можно найти в файлах `/etc/passwd` и `/etc/group`. Предположим, что ваш идентификационный номер пользователя на локальной машине равен 1000. На сервере Samba это значение, скорее всего, соответствует другому пользователю. Та же проблема может возникнуть и для идентификатора группы. С помощью выражений `uid=1001` и `gid=1001` вы сообщаете серверу Samba о своих данных на серверной машине. Эти значения надо найти в файлах `/etc/passwd` и `/etc/group`, но не на локальном компьютере, а на сервере Samba. Если вы не сделаете этого, вы создадите файлы и каталоги лишь для того, чтобы увидеть, что они не принадлежат вам и вы не можете использовать их.

И наконец, выражение `workgroup=TURING`, как нетрудно догадаться, определяет рабочую группу.

После монтирования разделяемого ресурса к нему можно обращаться с помощью любой программы. Например, вы можете запустить OpenOffice.org и непосредственно открыть файл в каталоге `eliot_documents/presentations/` или просмотреть PDF-файл в каталоге `eliot_documents/to_print` и вывести его содержимое на печать. Если вы по-

местите приведенные ниже строки в файл /etc/fstab, то разделяемый ресурс будет монтироваться автоматически.

```
//eliot/documents /home/scott/eliot_documents
smbfs credentials=/home/scott/credentials_eliot,f
mask=644,dmask=755,uid=1001,gid=1001,workgroup=TU
RING 0 0
```

Внимание!

Изменяя содержимое файла /etc/fstab, будьте очень осторожны! Ошибка в этом файле может привести к тому, что система перестанет загружаться. Конечно, данную проблему можно решить (рекомендации о том, как это сделать, вы найдете в моей книге *Hacking Knoppix*), но лучше не допускать ее возникновения. Информацию о файле fstab можно получить, выполнив команду man fstab.

Если вы решите, что вам больше не нужен доступ к разделяемому ресурсу Samba, смонтированному в каталоге eliot_documents, вы можете размонтировать его с помощью команды smbumount (обратите внимание: после символов sm следует umount, а не unmount).

```
$ smbumount eliot_documents
smbumount must be installed suid root
```

И снова ошибка. Команда smbumount также должна вызываться от имени пользователя root.

```
$ ls -l /usr/bin/smbumount
-rwxr-xr-x 1 root root /usr/bin/smbumount
$ sudo chmod u+s /usr/bin/smbumount
$ ls -l /usr/bin/smbumount
-rwsr-xr-x 1 root root /usr/bin/smbumount
```

Теперь только пользователь, который смонтировал ресурс в каталоге eliot_documents, может размонтировать данный ресурс (конечно, то же самое может сделать поль-

зователь root, поскольку его возможности в системе не ограничены).

```
$ smbumount eliot_documents
```

Вот и все. Если вы хотите убедиться, что команда smbumount выполнилась корректно, просмотрите содержимое каталога eliot_documents. Вы увидите, что он пуст. Соединение с разделяемым ресурсом Samba разорвано.

Выводы

Не секрет, что многие пользователи Linux недолюбливают продукты Microsoft. Тем не менее нельзя не признать, что сегодня мы живем в мире Windows. Samba обеспечивает "сосуществование" систем, позволяя пользователям Linux обращаться к ресурсам на машинах под управлением Windows и даже предоставляет ресурсы Linux пользователям других операционных систем. Samba — развитая, мощная, стабильно работающая система. Реализованная как проект с открытым исходным кодом, она постоянно развивается. Samba уменьшает зависимость от серверов Windows, и уже этим она хороша.

Предметный указатель

НЕОБХОДИМЫЙ КОД И КОМАНДЫ

Linux®

КАРМАННЫЙ СПРАВОЧНИК

В данной книге содержатся сведения, необходимые для эффективной работы с операционной системой Linux и средствами разработки программ.

ЛАКОНИЧНОСТЬ И ДОСТУПНОСТЬ

Эту книгу вы можете постоянно носить с собой и читать тогда, когда вам удобно. В ней кратко изложен материал, обычно содержащийся в толстых книгах.

ГИБКОСТЬ И ФУНКЦИОНАЛЬНОСТЬ

В этой книге вы найдете описания около 100 вариантов основных команд. Они позволят вам безотлагательно приступить к решению широкого круга задач.

Скотт Граннеман ведет ежемесячную рубрику в *SecurityFocus*; его работы также можно часто встретить в *Linux Magazine* и *The Open Source Weblog*. Он является адъюнкт-профессором Вашингтонского университета и читает различные курсы, связанные с информационными технологиями и глобальной сетью.

Операционные системы/Linux

Изображение на обложке © Medioimages/Getty Images

ISBN 978-5-8459-1118-6

www.williamspublishing.com

DEVELOPER'S LIBRARY
www.developers-library.com